НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

УДК: 004.42

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___” _____2019 р.

Дипломний проект
на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп'ютерні науки»*

на тему: «*Система підтримки синхронного дубляжу вистав*»

Виконав: студент *4* курсу, групи *ІС-51*
Кас`янчук Андрій Сергійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *доц., к.т.н., доц. Гриша О.В.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *доц., к.т.н., доц. Тєлишева Т.О.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *доц. каф. АУТС, к.т.н., доц. Репнікова Н.Б.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент Кас'янчук А.С. _____
(підпис)

Київ – 2019 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з п’яти розділів, містить 83 сторінки, 13 рисунків, 19 таблиць, 1 додаток, 4 джерела.

Дипломний проект присвячений розробці системи підтримки синхронного дубляжу вистав.

Метою комплексу задач є збільшення прибутку театру за рахунок підключення іноземних глядачів та надання можливості прослуховування вистав рідною мовою.

Програма дозволяє користувачам прослуховувати синхронний професійний дубляж вистави на своєму мобільному телефоні, використовуючи локальну мережу театру.

Були визначені вхідні та вихідні дані до системи, була розроблена структура бази даних для збереження інформації, яка відповідає поставленим цілям проекту.

Було описано та проведено порівняння методів. По результатам порівняння було обґрунтовано та обрано підхід імітаційного моделювання.

Описано основні засоби розробки системи, висунуті вимоги до технічного забезпечення та обрано архітектуру програмного забезпечення.

Призначенням розробки є підтримка процесу синхронізації дубляжу вистав декількох мов з використанням локальної мережі театру.

Розроблено інструкцію користувача та проведене тестування комплексу задач.

ТРАНСЛЯЦІЯ, ФАЙЛ, ПОТІК, БУФЕР, СИНХРОНІЗАЦІЯ, ДУБЛЯЖ, МЕРЕЖА, ФРАЗА.

ABSTRACT

Structure and scope of work. The explanatory note of the diploma project consists of five sections, containing 83 pages, 21 figures, 19 tables, 1 supplement, 6 sources.

The diploma project is devoted to the development of a system for supporting synchronous dubbing performances.

The aim of the set of tasks is to increase the profit of the theater by connecting foreign audiences and providing the opportunity to listen to plays in their own language.

Input and output data were identified in the system, a database structure was developed to store information that corresponds to the objectives of the project.

It was described and conducted a comparison of methods. The results of the comparison were substantiated and the simulation model was chosen.

The main means of system development, technical support requirements are outlined, software architecture is selected and substantiated.

A user manual has been developed and a set of tasks has been tested.

TRANSLATION, FILE, STREAM, BUFFER, SYNCHRONIZATION, DUBBING, NETWORK, SPEECH.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД	База даних
ООД	Об’єктно орієнтований дизайн
ООП	Об’єктно орієнтоване програмування
ОС	Операційна система
СУБД	Система управління базами даних
DTO	(Data Transfer Object) Об’єкт передачі даних
LAN	(Local area network) Локальна мережа
MVC	(Model view controller) Модель відображення контроллер
SOA	(Service-oriented architecture) Сервіс-орієнтована архітектура
WAN	(Wide area network) Глобальна мережа

ВСТУП

На планеті Земля живуть мільярди людей, які спілкуються на різних мовах. Тому було прийнято рішення розробити застосунок, який би руйнував мовний бар'єр в театрах. Ініціатором цього був Львівський театр ляльок. Кожен рік Львів відвідує велика кількість іноземних туристів, тому театр потенційно може збільшити кількість відвідувачів за рахунок людей, які не розуміють українську мову. Театр запропонував розробити застосунок, який би транслював професійний дубляж на телефон глядача. Глядач за допомогою своєї гарнітури, буде прослуховувати дубляж вистави. Так як театр – це жива дія, то стандартні методи дубляжу не підходять. Потрібно синхронізувати дію на сцені і переклад. Було запропоновано розробити клієнт-сторінку для адміністратора, на якій він міг би керувати виставою і синхронізувати її. Адміністратор має слідкувати за виставою і перемикати фрази на комп'ютері. Прив'язані до фрази аудіо-файли на різних мовах будуть транслюватись на телефон слухача, де попередньо було обрано мову прослуховування.

Спочатку було розглянуто аналоги або сервіси, завдяки яким можна було б вирішити дану задачу. Готові рішення не підходили через велику затримку і через великі витрати коштів для інтеграції, тому було прийняти розробляти систему з нуля. Було розроблений алгоритм транслювання аудіо-файлів, при якому затримка була менше 0,5 секунди та розроблено веб-сайти для користувачів і адміністратора.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Українські театри потенційно можуть відвідувати туристи з інших країн, але головною проблемою є те, що люди які не розуміють українську мову, не можуть зрозуміти виставу. Існує декілька варіанті вирішення цієї проблеми.

Можна проводити вистави на різних мовах, але в даному варіанті дуже багато мінусів. Актори повинні відмінно знати іноземну мову та, навряд чи, вистава в Україні на не українській або російській мові збере багато глядачів. Даний варіант можна відразу відкинути, так як він має більше недоліків ніж переваг.

Другим варіантом є транслювання перекладу на екран у вигляді біжучого рядка. Даний спосіб не має недоліків попереднього варіанту, але має свої серйозні проблеми. Глядачі будуть не дивитись виставу, а відволікатись на біжучий рядок. Плюс до того, для реалізації даного варіанту потрібен великий екран або проектор, за допомогою яких буде відображатись текст, коштують дорого.

Найкращим варіантом є транслювання аудіо дубляжу в вухо глядача. Це можна реалізувати за допомогою Web-застосунку, який буде транслювати аудіо з сервера на телефон клієнта, після чого глядач, за допомогою навушників, зможе слухати переклад вистави. Цей спосіб не має недоліків попередніх варіантів, тому будемо використовувати даний спосіб вирішення проблеми.

Транслювання дубляжу виставу має низку проблем. Вистава це не фільм, і тому підготувати один аудіо файл заздалегідь не вийде, тому що театр – це жива гра акторів, а актори – це люди, які можуть помилитись і вистава піде не по плану. Для вирішення цієї проблеми можна розділити підготовлений аудіо-файл дубляжу на частинки по 10-20 секунд і керувати

ними. Аудіо-файлами керуватиме оператор, спостерігаючи за виставою, буде перемикати фрази вистави, за якими прив'язаний аудіо файл дубляжу обраної мови.

Існує багато способів транслювання аудіо, але важливо обрати спосіб з мінімальною затримкою. Класичні способи транслювання (інтернет-радіо і тд.) мають досить затримку 5-10 секунд, що для нашого випадку є занадто багато, тому потрібно досягти затримки менше однієї секунди.

За допомогою Web-застосунку можна збирати статистику про кількість підключених користувачів до сервісу, обрану мови дубляжу та отримувати відгуки від користувачів для аналізу.

Даний веб застосунок можливо розмістити або на віддаленому сервері або на локальній машині, що допоможе зберегти гроші на оренду серверних потужностей і зменшить затримку трансляції.

1.1.1 Опис процесу діяльності

Розглянемо дії, які має виконати користувач для створення вистав і керування трансляцією. Схему структурну діяльності наведено в частині графічного матеріалу.

1.1.2 Опис функціональної моделі

Для проектування діаграми використання спочатку необхідно визначити дійових осіб (акторів), а потім визначити, які дії у системі може виконувати кожен з акторів. Приведемо акторів системи:

– слухач;
– адміністратор.

Нижче наведений опис кожного з акторів.

Слухач. Відвідувач театру, який має змогу обрати виставу, мову дубляжу і слухати дубляж вистави. Після завершення вистави може відвідати сторінки соціальних мереж театру.

Адміністратор. Працівник театру, що може починати трансляцію і керувати нею. А також створювати, редагувати вистави та завантажувати аудіо файли.

Тепер визначимо дії, які можуть виконувати актори системи. Схему структурну варіантів використання наведено в частині графічного матеріалу.

Розпишемо детальніше деякі з варіантів використань:

- обрати виставу і мову для прослуховування дубляжу вистави:
  1) вибір вистави – обирання потрібної вистави серед доступних;
  2) вибір мови дубляжу – обирання мови на якій клієнт буде прослуховувати дубляж вистави;
  3) прослуховування дубляжу – транслювання дубляжу вистави на пристрій.
- керування мовами дубляжу:
  1) створення мови дубляжу – створення мови до якої будуть прив’язуватись аудіо файли;
  2) редагування мови дубляжу – змніна назви мови;
  3) видалення мови дубляжу – видалення мови дубляжу, при якому видаляються усі аудіофайли прив’язані до цієї мови.
- керування виставами:
  1) створення вистави – заповнення форми з назвою і описом вистави, добавлення фраз вистави і завантаження до них аудіо;
  2) редагування вистави – зміна назви і опису вистави, видалення або редагування фраз та перезавантаження аудіо файлів;

3) керувати трансляцією – транслювання аудіо-файли на пристрої клієнтів та керування (перемикання фраз, пауза) трансляцією.

1.2 Огляд наявних аналогів

Під час розробки даного рішення було проаналізовано різні способи транслювання аудіо потоку. Популярний спосіб транслювання аудіо, такі як онлайн-радіо не зовсім підходив під вимоги. Транслювання цим способом має досить велику затримку (3-5 сек.), що для живого перегляду є критично. Було прийнято рішення транслювати аудіо методом пребуфферизації, так як нам відомо, що буде грати в ефірі в майбутньому.

В ході пошуку схожих вирішень було виявлено підходів зі схожою функціональністю:

- онлайн-радіо;
- Shoutcast від компанії Nullsoft;
- Spotify.

Всі ці підходи мають велику затримку трансляції, що для нашої предметної області є недопустимо. Також в цих підходах є проблеми з завантаженням аудіо файлів і транслюванням одночасно декількох потоків на декількох мовах. Характеристики підходів наведені в таблиці 1.1.

Таблиця 1.1 - Характеристика існуючих підходів

Назва	Опис
Онлайн-радіо	Стандартна технологія транслювання аудіо потоку через мережу Інтернет. Використовується більшістю інтернет-радіостанціями, має затримку 3-5 секунд
Shoutcast від компанії Nullsoft	SHOUTcast DNAS - це крос-платформне програмне забезпечення для потокової передачі мультимедійних даних через Інтернет. Програмне забезпечення,

Продовження таблиці 1.1

Назва	Опис
	розроблене компанією Nullsoft, доступне безкоштовно. Це дозволяє цифровому аудіоконтенту, в першу чергу у форматі MP3 або High-Efficiency Advanced Audio Coding, транслюватись до програмного забезпечення медіаплеєра та створювати інтернет-радіостанції. Найбільш поширене використання SHOUTcast для створення або прослуховування аудіо трансляції в Інтернеті, однак існують також відео потоки. Деякі традиційні радіостанції використовують SHOUTcast для розширення своєї присутності в Інтернеті. Цей сервіс міг би допомогти реалізувати дану задачу, але так як цей сервіс базується на технології онлайн радіо, то він має відповідну затримку, що є критично
Spotify	онлайн-сервіс потокового аудіо, який дає змогу безкоштовно прослуховувати музичні композиції. Дозволяє легального онлайнового стримінгу аудіозаписів. Дає змогу транслювати свої аудіозаписи

1.3 Постановка задачі

1.3.1 Призначення розробки

Призначенням розробки є підтримка процесу синхронізації дубляжу вистав декількох мов з використанням локальної мережі театру.

1.3.2 Цілі та задачі розробки

Метою комплексу задач є збільшення прибутку театру за рахунок підключення іноземних глядачів та надання можливості прослуховування

вистав рідною мовою. Для досягнення поставленої мети мають бути вирішені такі задачі:

- ведення мов;
- ведення вистав;
- ведення фраз вистави;
- підключення користувача до системи;
- ініціація початку та завершення трансляції;
- активізація фраз.

Висновок до розділу

В даному розділі було виконано пошук існуючих способів вирішення проблем. Проведено аналіз переваг та недоліків існуючих аналогів. Було визначено призначення роботи, цілі та задачі розробки.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Вхідні дані комплексу задач транслювання аудіо-потоку надходять з декількох джерел, а саме від:

- користувачів (глядачів театру);
- адміністратора.

Тепер детально розглянемо, які саме дані надходять з цих джерел.

Дані, які надходять від користувачів. Перед початком прослуховування дубляжу, користувач має повідомити наступні дані:

- назва вистави;
- мова дубляжу.

Дані, які надходять від адміністратора. При керуванні виставами адміністратор має повідомити наступні дані:

- назва вистави;
- опис вистави;
- список фраз;
- список мов;
- аудіо-файли до фраз.

Всі дані зберігаються в базі даних, крім файлів. Файли зберігаються в файловій системі зі спеціальною назвою, по якій пізніше проходить пошук цього файлу. Інформація про файл зберігається в базі даних.

2.2 Вихідні дані

Вихідними даними є список вистав. Список вистав представляється наступними даними:

- назва вистави;

- опис вистави;
- список фраз;
- список аудіо-файлів до кожної фрази.

Всі дані зберігаються в базі даних крім аудіо-файлів. Аудіо-файли зберігаються в файловій системі сервера.

2.3 Опис структури бази даних

База даних використовується для збереження ы

У таблиці 2.1 наведено опис таблиць розробленої бази даних.

Таблиця 2.1 – Таблиця клієнтів

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
Performances – таблиця вистав	Id	numeric	Код, під яким вистава зберігається в базі даних
	Titel	text	Назва вистави
	Description	text	Опис вистави
Speeches – таблиця фраз до вистав	Id	numeric	Код, під яким фраза зберігається в базі даних
	Order	numeric	Порядок фрази в виставі
	Text	text	Текст фрази
	Duration	numeric	Час програвання фрази (вираховується як тривалість найтривалішого аудіо поточної фрази)
	PerformanceId	numeric	Код вистави
Audio – таблиця з інформацією про аудіо	Id	numeric	Код, під яким інформація про аудіо зберігається в базі даних

Продовження таблиці 2.1

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
	FileName	text	Назва файлу, збереженого в файлову систему
	OriginalText	text	Текст фрази (потрібен для завантаження файлів пакетом)
	Duration	numeric	Тривалість аудіо
	SpeechId	numeric	Код фрази
	LanguageId	numeric	Код мови
Languages – таблиця з мовами	Id	numeric	Код, під яким фраза зберігається в базі даних
	Name	text	Назва мови

Схему структурна бази даних наведена в частині графічного матеріалу.

Висновок до розділу

В даному розділі було визначено вхідні та вихідні дані. Було описано та спроектовано базу даних.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Працівники театру перед виставою повинні перевірити наявність білету. Через це перед входом в зал театру накопичується черга. Театр хотів би розрахувати оптимальну кількість працівників театру, які будуть перевіряти наявність білету. Заздалегідь відомо інтенсивність приходу глядачів та кількість місць в залі. Потрібно розрахувати таку кількість пропускних пунктів, щоб довжина черги була не більше n.

3.2 Математична постановка задачі

Призначенням цієї задачі є знаходження оптимальної кількості пропускних пунктів, щоб максимальна довжина черги була менше n.

Дано:

- Інтенсивність приходу глядачів в театр (задається у вигляді функції);
- Максимальна допустима довжина черги – *qmax*;
- Витрати на одній пропускний пункт – *p*;
- Поточна довжина черги *i*-го пропускного пункту – q_i.

Змінні:

- Кількість пропускних пунктів – n.

Цільова функція – мінімізувати витрати на пропускні пункти:

$$n * p \rightarrow min \quad (3.1)$$

Обмеження:

$$q_i \leq qmax \quad (3.2)$$

3.3 Обґрунтування методу розв’язання

Для вирішення проблеми можна використовувати різні методи. Давайте подивимося на деякі з них і виберемо потрібний.

Основні умови вибору методу:

- постановка завдання;
- складом, характером та обсягом вхідних даних;
- часом на рішення дослідницької задачі.

Природні або фізичні випробування (дослідження). Вимірювання характеристик здійснюється на досліджуваних системах в режимі реального часу (проведення експериментів); дані отримані дослідником шляхом моніторингу процесів в реальній системі.

Переваги:

- висока адекватність моделі реальній системі;
- висока точність результатів.

Недоліки:

- висока вартість створення моделі;
- високі часові витрати;
- необхідність доопрацювання окремих вузлів реальної системи для проведення натурних експериментів.

Аналітичне моделювання. Модель представлена набором аналітичних виразів, які відображають явні функціональні залежності між параметрами реальної системи при її експлуатації: лінійні та нелінійні рівняння, диференціальні та інтегральні рівняння, імовірнісні залежності; Аналітичні моделі використовуються для відносно простих систем, для вивчення характеристик, які не вимагають високої точності. Тому аналітична модель є грубим наближенням до дійсності.

Переваги:

- висока точність обчислень;
- простота і низька вартість моделі;
- можливість швидко отримати результати.

Недоліки:

- велике число припущень;
- велике число обмежень;
- невисока точність результатів;
- відповідність результатів певним умовам;
- складність аналітичного опису.

Імітаційне моделювання. Суть імітаційного моделювання полягає в імітації процесу функціонування системи в часі, з дотриманням таких же співвідношень тривалості операцій як в системі оригіналі. При цьому імітуються елементарні явища, що становлять процес; зберігається їх логічна структура, послідовність протікання в часі. Результатом імітаційного моделювання є отримання оцінок характеристик системи.

Переваги:

- висока адекватність між фізичною суттю описуваного процесу та його моделлю;
- можливість описати складну систему на досить високому рівні деталізації;
- значно більше областей дослідження, ніж для аналітичного моделювання;
- відсутність обмежень відображення в моделі залежностей між параметрами моделі;
- можливість оцінки функціонування системи не тільки в стаціонарних станах, але і в перехідних режимах (процесах);
- одержання значної кількості даних про досліджуваний об'єкт (закон розподілу випадкових величин, числові значення абсолютні та відносні, і багато іншого);

- найбільш раціональне ставлення «результат - витрати» по відношенню до аналітичного і фізичного моделюванню.

Недоліки:

- розробка хорошої моделі часто обходиться дорожче, ніж аналітична і вимагає більше часу на створення і налагодження;
- складно оцінити ступінь точності моделі, її адекватність досліджуваному процесу;
- відносно високі вимоги до кваліфікації дослідника для написання моделі;
- спільність застосування та індивідуальність реалізації.

Комбіновані методи моделювання. Модель представляється у комбінації методів моделювання; найбільш широко застосовуються імітаційно-аналітичні моделі; ступінь застосування методів моделювання визначає дослідник, виходячи з поставлених завдань, наявних ресурсів (знань, комп'ютера) і часу на проведення дослідницької роботи.

Кращим для вирішення поставленої задачі буде метод імітаційного моделювання тому, що в системі моделювання присутній час, а це робить недоцільним використання аналітичного моделювання, а вартість експериментів коштує дуже велику кількість грошей, а це робить недоцільним використання експериментального методу. Реалізація імітаційної моделі відбувається за допомогою мови моделювання GPSS.

3.4 Опис методів розв’язання

Задачу буде вирішено методом імітаційного моделювання.

Змоделюємо систему з n пристроями (пропускними пунктами). В систему транзакти будуть поступати з заданою інтенсивністю і будуть потрапляти в чергу, звідки потрапляють до пристрою. Для моделювання системи було використано мову GPSS і платформу GPSS World. Код програми в лістингу 3.1.

Лістинг коду 3.1

```
EXPONENT FUNCTION AC1,C10
0,8/10,7/20,7/30,3/40,2/50,1/60,2/70,3/80,8/90,9


checks    STORAGE 4

          GENERATE  FN$EXPONENT
          QUEUE     queue1
          ENTER     checks
          ADVANCE   3
          DEPART    queue1
          LEAVE     checks

          TERMINATE
          GENERATE  90
          TERMINATE 1
            START 1
```

Проаналізувавши результати з різним n, знайдемо відповідь на поставлену задачу.

3.5 Аналіз випробувань

Проведемо випробування для знаходження оптимальної кількості пропускних пунктів. Максимальна довжина черги має бути не більше 3-ьох.

Інтенсивність задамо функцією зображеною на рисунку 3.1. Вісь x задає час до початку вистави помножений на -1.

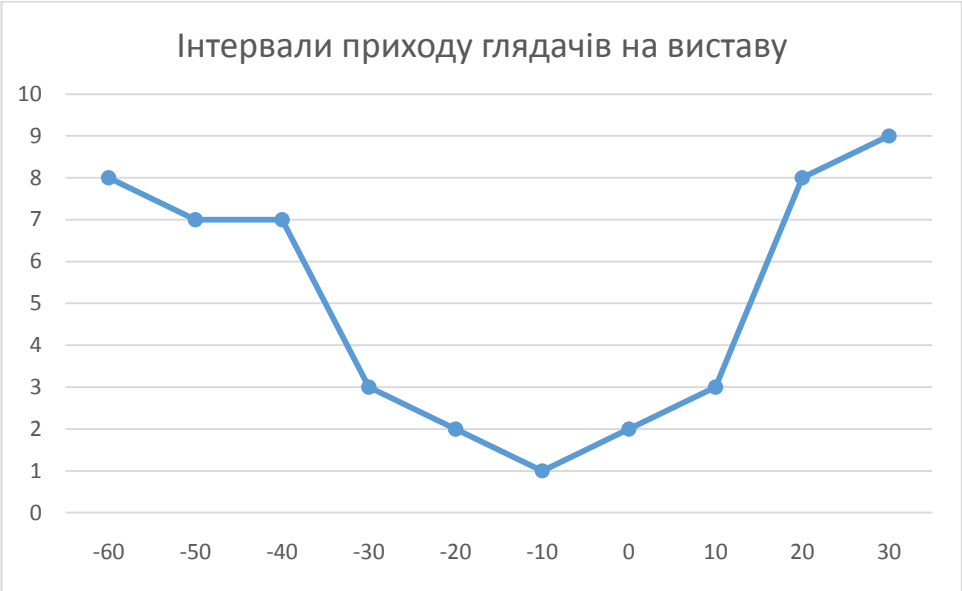


Рисунок 3.1 – Функція інтервалів

У таблиці 3.1 наведено опис експериментів.

Таблиця 3.1 – Таблиця експериментів

№	Кількість пропускних пунктів	Максимальна довжина черги	Кількість експериментів
1	1	10	50
2	2	4	50
3	3	3	50
4	4	3	50

За результатами експериментів зрозуміло, що оптимальною кількістю пропускних пунктів є 3.

Висновок до розділу

В даному розділі було описано змістовну і математичну постановку задачі. Проведено обґрунтування методів розв’язання. Описано обраний метод розв’язання задачі та проведено аналіз результатів.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

При створенні програмного продукту були використанні оболонки Visual Studio 2017 для програмування на мові C#, Visual Studio Code для програмування на мові TypeScript та CSS та SQLite Studio для керування базою даних SQLite.

Мову програмування C# було обрано через приналежність до платформи .NET Core. .NET Core є мультиплатформенним, що дозволяє розробляти застосунки для операційних систем Windows, Linux та macOS, на відміну від .NET, що підтримувала лише Windows. В платформу .NET Core входить технологія розробки веб-застосунків ASP.NET, яка буде використовуватись для розробки нашого web-застосунку. За допомогою ASP.NET можна розробити WEB API застосунок, який буде легко масштабуватись, підтримуватись і модифікуватись. У випадку ASP.NET, веб-API - це простий спосіб реалізації веб-служб RESTful застосунків за допомогою .NET framework. Веб-сервіси RESTful - це ті, що використовують HTTP як основний метод комунікації. Веб-API ASP.NET в основному визначається як фреймворк, який дозволяє розробці HTTP-служб охопити об'єкти клієнта, такі як браузери, пристрої або планшети. ASP.NET Web API можна використовувати з MVC для будь-якого типу додатків. Таким чином, .NET веб-API дуже важливі для розробки веб-додатків ASP.NET.

Також платформа .NET Core підтримує фреймворк SignalR, який потрібен для реалізації транслювання аудіо-потоку. ASP.NET SignalR - це бібліотека для розробників ASP.NET, що спрощує процес додавання веб-функціональних можливостей у реальному часі до додатків. Веб-функціональність у режимі реального часу - це можливість мати серверний код, який набирає контент для підключених клієнтів, як тільки він стане доступним, замість того, щоб сервер чекав, коли клієнт запитає нові дані. SignalR може використовуватися для додавання будь-якого типу "реального

часу" веб-функціональності до вашої програми ASP.NET. Хоча чат часто використовується як приклад, ви можете зробити набагато більше. Кожного разу, коли користувач оновлює веб-сторінку, щоб побачити нові дані, або сторінка впроваджує довгі опитування для отримання нових даних, вона є кандидатом для використання SignalR. Приклади включають приладові панелі та програми моніторингу, спільні програми (наприклад, одночасне редагування документів), оновлення прогресу роботи та форми в реальному часі.

C# підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Не менш важливою перевагою мови C# є схожість з мовами Java, C++, що дозволить програмістам, які знають синтаксис попередньо вказаних мов, легко освоїтись.

Для збереження даних було обрано базу даних SQLite. Головною перевагою цієї бази даних є її простота і те що вона базується на файлах. Так як сервер буде розміщуватись на персональному комп'ютері, то дуже важливо, щоб базу даних було легко встановлювати. У випадку SQLite нічого додаткового встановлювати не потрібно, програма сама при першому старті програми створить файл бази даних. Так як в SQLite база даних, то з легкістю можна робити копії бази даних для відновлення, у випадку непередбачуваних помилок. SQLite прекрасно працює як двигун бази даних для більшості веб-сайтів з низьким або середнім рівнем трафіку (тобто більшість веб-сайтів). Обсяг веб-трафіку, який може обробляти SQLite, залежить від того, наскільки сайт використовує свою базу даних. Взагалі кажучи, будь-який сайт, який отримує менше, ніж 100K хітів / день, повинен добре працювати з SQLite. Цифри в 100K / день є консервативною оцінкою, а не жорсткою верхньою межею. Показано, що SQLite працює з 10-кратним обсягом трафіку.

Для розробки клієнтської частини веб-застосунку було обрано мову TypeScript та бібліотека для розробки SPA (single-page aplication) React. Перевагами TypeScript над JavaScript є можливість явного визначення типів

(статична типізація), підтримка використання повноцінних класів (як в традиційних об'єктно-орієнтованих мовах), підтримка підключення модулів. React це бібліотека для створення користувацьких інтерфейсів. Однією з її характерних особливостей є можливість використовувати TSX, мова програмування з близьким до HTML синтаксисом, який компілюється в TypeScript. Розробники можуть вимагати більшої продуктивності додатків за допомогою Virtual DOM. З React можна створювати ізоморфні додатки, які допоможуть позбавитися від неприємної ситуації, коли користувач з нетерпінням чекає, коли завершиться завантаження даних і на екрані його комп'ютера нарешті з'явиться щось крім анімації завантаження. Створені компоненти можуть бути з легкістю змінені і використані заново в нових проектах.

Back-end частина розроблялася на алтформі Microsoft Visual Studio. Visual Studio включає в себе редактор коду, що підтримує IntelliSense (компонент завершення коду), а також рефакторинг коду. Інтегрований відладчик працює як відладчик вихідного рівня, так і відладчик на рівні машини. Інші вбудовані інструменти включають в себе профайлер коду, дизайнер форм для створення GUI-додатків, веб-дизайнер, дизайнер класів і дизайнер схеми бази даних. Він приймає плагіни, що підвищують функціональність практично на кожному рівні, включаючи додавання підтримки систем керування джерелами (наприклад, Subversion і Git) і додавання нових наборів інструментів, таких як редактори та візуальні дизайнери для доменних мов або наборів інструментів для інших аспектів розробки програмного забезпечення (наприклад, клієнт Team Foundation Server: Team Explorer).

Front-end частина розроблялася на кроссплатформенному редакторі коду Visual Studio Code. У своєму сервісі Visual Studio Code має блискавичний редактор вихідних кодів, ідеально підходить для щоденного використання. Завдяки підтримці сотень мов, VS Code допомагає вам бути миттєво продуктивними з підсвічуванням синтаксису, відповідності дужок,

автоматичного відступу, вибору вікон, фрагментів і багато іншого. Інтуїтивно зрозумілі комбінації клавіш, легкі налаштування та доповнення клавіатурних скорочень за допомогою спільноти дозволяють легко переміщатися по коду.

4.2 Вимоги до технічного забезпечення

4.2.1 Загальні вимоги

Програмний продукт являє собою комплекс серверної частини і клієнтської частини. Серверна і клієнтська частина буде розгортатись на комп’ютері в локальній мережі. Клієнтська частина буде доступна з локальної мережі.

Для правильної роботи даної програми до складу технічних засобів повинні входити:

- комп’ютер з такою конфігурацією:
    1) процесор з тактовою частотою не нижче 2 ГГц;
    2) достатній об’єм оперативної пам’яті (не менше 4 ГБ);
    3) інші складові можуть мати будь-які параметри, тому що вони не значним чином впливають на роботу програми.
- додатково має бути встановлене таке програмне забезпечення:
    1) операційна система Windows 7/8/10;
    2) .NET Core 2.2 і вище;
    3) Node.js;
    4) Браузер Chrome, Firefox, Opera.
- комп’ютерна периферія, до складу якої входить:
    1) монітор;
    2) мишка;
    3) клавіатура;
    4) Wi-Fi роутер.

4.2.2 Опис локальної обчислювальної мережі

Локальна мережа (LAN) - це комп'ютерна мережа, яка з'єднує комп'ютери в межах обмежених територій, таких як резиденція, школа, лабораторія, університетський кампус або офісна будівля. На відміну від цього, широкосмугова мережа (WAN) не тільки охоплює більшу географічну відстань, але також, як правило, включає в себе орендовані телекомунікаційні схеми. Локальна мереже дає змогу скоротити затримки при транслюванні аудіо файлів.

Мережа складається з комп'ютера, Wi-Fi роутера і телефонів відвідувачів. На комп’ютері буде розміщуватись сервер. З цього або ж іншого комп’ютера в мережі буде проходити керування виставою. Структура мережі зображена на рисунку 4.1.

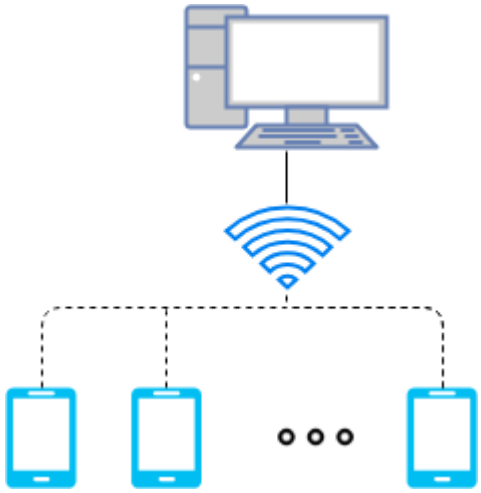


Рисунок 4.1 – Структура мережі

4.3 Архітектура програмного забезпечення

Доцільна побудова наступних діаграм UML фази уточнення:

- Діаграма послідовності — діаграма, що визначає послідовність взаємодії об’єктів класів для вирішення задачі;
- Діаграма класів — діаграма, що показує набір класів та їх взаємодії для вирішення задачі.

4.3.1 Діаграма послідовності

На діаграмі задіяні наступні класи. Схему структурну послідовності наведено в частині графічного матеріалу. У таблиці 4.1 вказані задіяні класи та їх опис.

Таблиця 4.1 – Таблиця задіяних класів

Клас	*Відповідальність*
Адміністратор	Введення назви і опису вистави, додавання фраз да завантаження до них аудіо-файлів
Інтерфейс	Відображення форми та відображення даних
Сервер	Приймання, передавання та обробка інформації
БД	Збереження інформації про вистави та фрази до вистав
Файлова система	Збереження аудіо-файлів та перейменування їх

4.3.2 Діаграма класів

Діаграми класів відображення сутностей

Створимо діаграму класів для сутностей засобами Visual Studio 2017 Community. Для цього, в проекті створимо файл (рисунок 4.3).

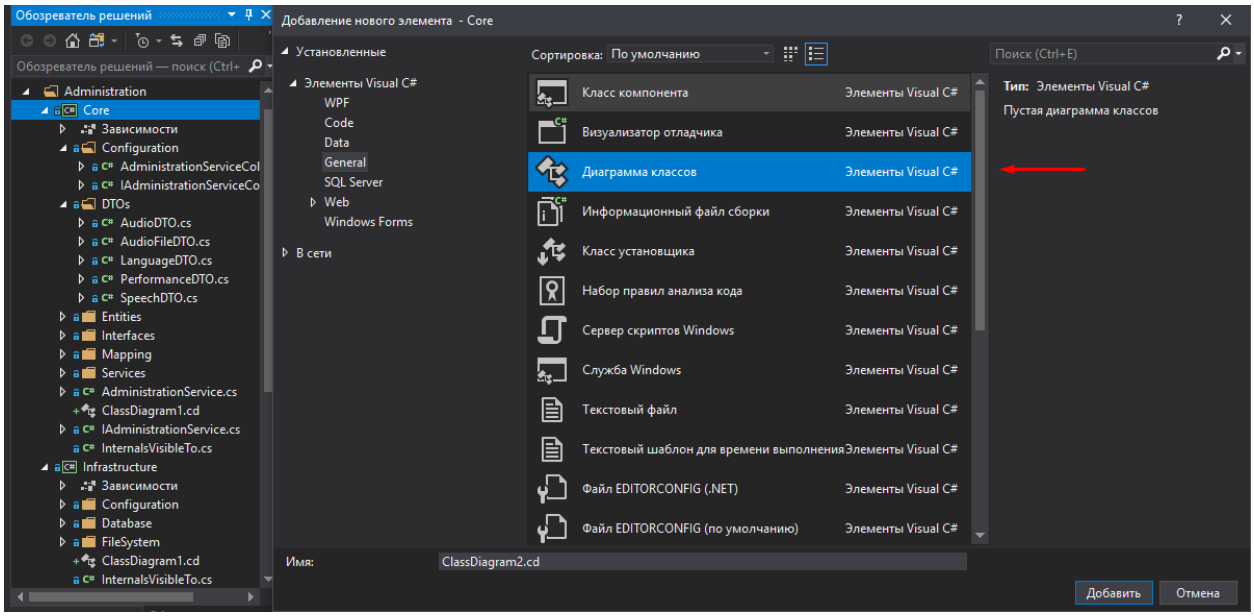


Рисунок.4.3 – Вікно додавання нового елементу

В відкрите вікно перенесемо потрібні класи і отримаємо діаграму класів для сутностей. Сутності описують об’єкти, з якими ми будемо працювати в базі даних за допомогою Entity Framework Core. Entity Framework являє спеціальну об'єктно-орієнтовану технологію на базі фреймворка .NET для роботи з даними. Якщо традиційні засоби ADO.NET дозволяють створювати підключення, команди та інші об'єкти для взаємодії з базами даних, то Entity Framework являє собою більш високий рівень абстракції, який дозволяє абстрагуватися від самої бази даних і працювати з даними незалежно від типу сховища. Якщо на фізичному рівні ми оперуємо таблицями, індексами, первинними і зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працює з об'єктами. Схему структурну класів наведено в частині графічного матеріалу.

Діаграма містить 5 класів, а саме:

- «BaseEntity» – клас, від якого наслідуються усі сутності;
- «Audio» – клас аудіо, який має поля для роботи з аудіо;
- «Performance» – клас вистави, який має поля для роботи з виставами;
- «Language» – клас мови, який має поля для роботи з мовами;
- «Speech» – класс фрази, який має поля для роботи з фразами.

Діаграми класів для відображення DTOs (data transfer object)

Створимо діаграму класів для відображення класів, які потрібні для передачі об’єктів між клієнтом і сервером та навпаки. Об'єкт передачі даних (DTO) є об'єктом, який переносить дані між процесами. Мотивацією його використання є те, що зв'язок між процесами зазвичай здійснюється з використанням віддалених інтерфейсів (наприклад, веб-служб), де кожен виклик є дорогою операцією. Оскільки більшість витрат на кожний виклик пов'язана з тимчасовим часом між клієнтом і сервером, одним із способів зменшення кількості викликів є використання об'єкта (DTO), який агрегує

дані, які були б передані кількома дзвінками, але це відбувається тільки одним викликом. Схему структурну класів наведено в частині графічного матеріалу.

Діаграма містить 5 класів:

– «AudioDTO» – клас для трансферу об’єктів типу Audio;
– «AudioFileDTO» – клас для трансферу аудіо-файлів;
– «LanguageDTO» – клас для трансферу об’єктів типу Language;
– «PerformanceDTO» – клас для трансферу об’єктів типу Performance;
– «SpeechDTO» – клас для трансферу об’єктів типу Speech.

Приклад діаграми класів для відображення сервісів

Створимо діаграму класів для відображення сервісів, які відповідають за логіку програми. Схему структурну класів наведено в частині графічного матеріалу.

Діаграма містить 9 класів:

– «GenericService» – узагальнений клас, який реалізує всі основні CRUD (create-read-update-delete) операції;
– «AudioService» – клас, який наслідується від класу Generic Service і реалізовує методи для роботи з Audio;
– «PerformanceService» – клас, який наслідується від класу Generic Service і реалізовує методи для роботи з Performance;
– «SpeechService» – клас, який наслідується від класу Generic Service і реалізовує методи для роботи з Speech;
– «LanguageService» – клас, який наслідується від класу Generic Service і реалізовує методи для роботи з Language;
– «DubbingContext» – клас контексту бази даних;
– «Repository» – клас, який реалізує методи комунікації з базою даних;
– «FileSystemRepository» – клас, який реалізовує методи комунікації з файловою системою.
– «AdministrationService» – клас, який реєструє усі методи сервісів;

4.3.3 Специфікація функцій

Класи представляють набір сервісів. Сервіс-орієнтована архітектура (SOA) - це стиль проектування програмного забезпечення, де послуги надаються іншим компонентам компонентами програми, через протокол зв'язку через мережу. Основні принципи сервісно-орієнтованої архітектури не залежать від постачальників, продуктів і технологій. Послуга є дискретною одиницею функціональних можливостей, до якої можна отримати віддалений доступ і діяти і оновлюватись самостійно, наприклад, витяг онлайн-виписки з кредитної картки. Різні послуги можуть використовуватися спільно для забезпечення функціональності великого програмного забезпечення, принцип SOA поділяється з модульним програмуванням. Сервіс-орієнтована архітектура об'єднує розподілені, окремо підтримувані та розгорнуті компоненти програмного забезпечення. Це забезпечується технологіями та стандартами, які полегшують зв'язок компонентів та співпрацюють через мережу, особливо через мережу IP.

Функції класів програмного забезпечення наведені в таблиці 4.2.

Таблиця 4.2 – Функції класів програмного забезпечення

Назва	**Примітка**
Клас: GenericService – узагальнений клас, від якого унаслідуються усі сервіси. Реалізовує основні CRUD операції	
public virtual async Task<List<T>> GetAllAsync()	Повертає список усіх T
public virtual async Task<T> GetByIdAsync(int id)	Повертає об’єкт T з id
public virtual async Task CreateAsync(T entity)	Зберігає в базі даних об’єкт
public virtual async Task UpdateAsync(int id, T newEntity)	Оновлює в базі даних об’єкт

Назва	Примітка
public virtual async Task DeleteAsync(int id)	Видаляє об’єкт з бази даних
Клас: AudiService – клас, який унаслідується від GenericService<Audio> та реалізовує функціонал для роботи з аудіо	
public override async Task CreateAsync(Audio entity)	Створює запис інформації про аудіо-файл в базі даних
public override async Task UpdateAsync(int id, Audio newEntity)	Оновлює запису про аудіо-файл в базі даних і перейменування файлу в файловій системі
public override async Task DeleteAsync(int id)	Видаляє всі аудіо-файли фрази з кодом id
public async Task DeleteFileAsync(int id)	Видаляє файл запису про аудіо-файл р базі даних
public void DeleteAudioFiles(IEnumerable<string> namesList)	Видаляє масив аудіо-файлів
private async Task<Audio> ChangeName(Audio entity)	Перейменовує файл
private async Task<Audio> ChangeDuration(Audio entity)	Змінює тривалість аудіо-файлу в базі даних
Клас: LanguageService – клас, який унаслідується від GenericService<Language> та реалізовує функціонал для роботи з мовами дубляжу	
public override async Task DeleteAsync(int id)	Видаляє мову та всі аудіо-файли на цій мові
Клас: PerformanceService – клас, який унаслідується від GenericService<Performance> та реалізовує функціонал для роботи з виставами	
public async Task<List<Speech>> GetChildrenByIdAsync(int id)	Повертає список фраз вистави з кодом id
public override async Task DeleteAsync(int id)	Видаляє виставу та фрази цієї вистави

Продовження таблиці 4.2

Назва	Примітка
public async Task<List<Language>> GetLanguagesAsync(int id)	Повертає список мов дубляжа цієї вистави
Клас: SpeechService – клас, який унаслідується від GenericService<Speech> та реалізовує функціонал для роботи з фразами	
public async Task<List<Audio>> GetChildrenByIdAsync(int id)	Повертає список усіх аудіо-файлів фрази з кодом id
public override async Task DeleteAsync(int id)	Видаляє фразу та ауді-файли цієї фрази

Висновок до розділу

В даному розділі було розглянуто засоби розробки та проведено порівняння існуючих платформ для розробки. Було задано загальні вимоги до системи та описано локальну обчислювальну мережу. Було розглянуто архітектуру системи, побудовано діаграми класів та діаграму послідовності.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Для початку роботи з застосування потрібно відкрити браузер і перейти по адресі http://localhost:3000. Головна сторінка сайту містить кнопки до переходу на сторінки керування мовами дубляжу, створення вистав та список вистав. Головна сторінка з позначками зображена на рисунку 5.1.

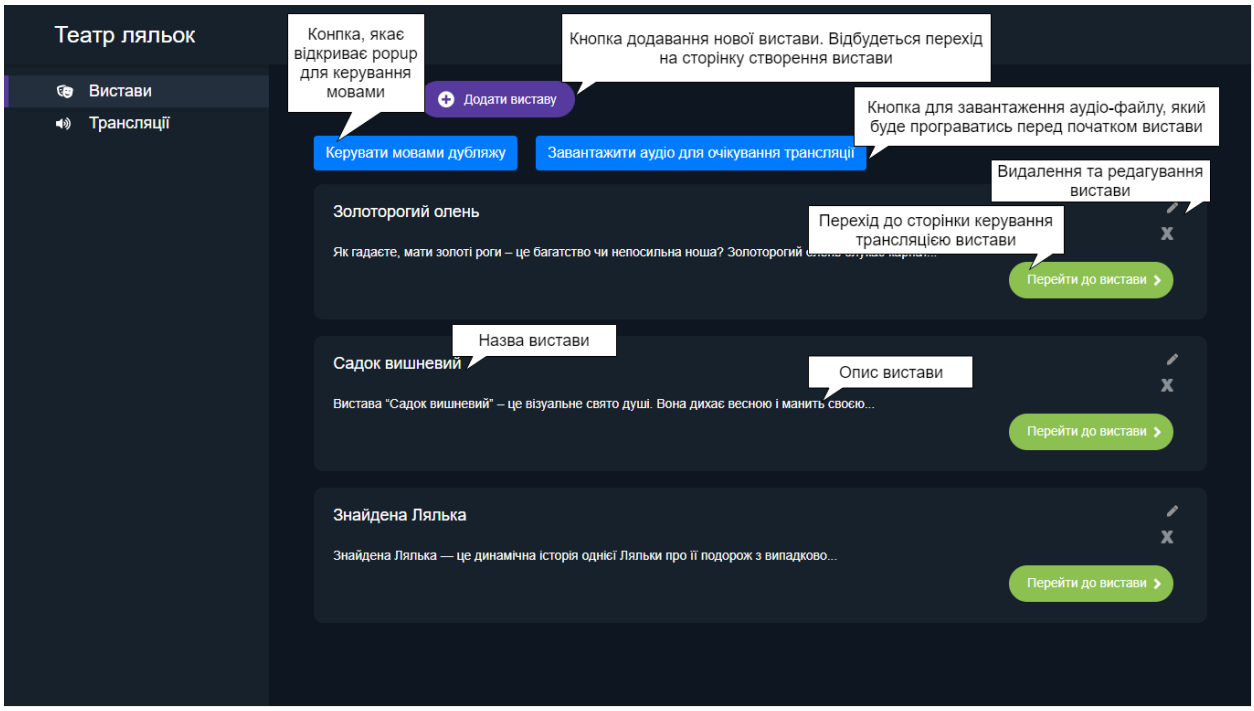


Рисунок 5.1 – Головна сторінка програмного застосування

Спочатку потрібно добавити мови дубляжу, до яких будуть прив’язуватись аудіо-файли. Для цього потрібно натиснути на кнопку «Керувати мовами дубляжу», після чого відкриється спливаюче вікно (рисунок 5.2).

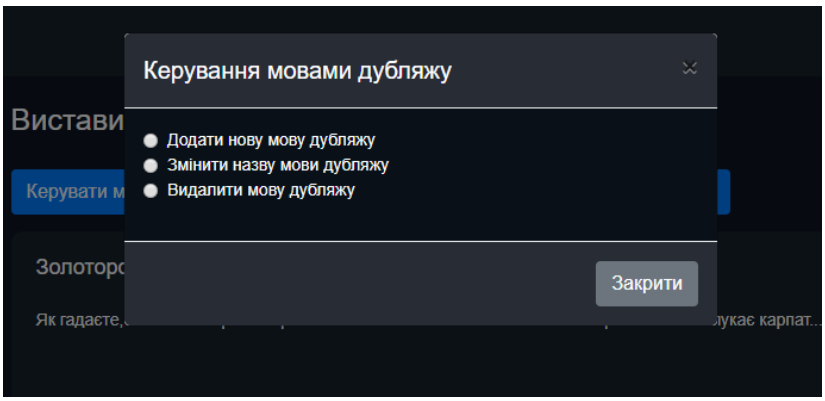


Рисунок 5.2 – Спливаюче вікно керування мовами дубляжу

Для додавання мови потрібно вибрати пункт «Додати нову мову дубляжу», вести назву мови в поле та натиснути кнопку «Додати мову» (рисунок 5.3).

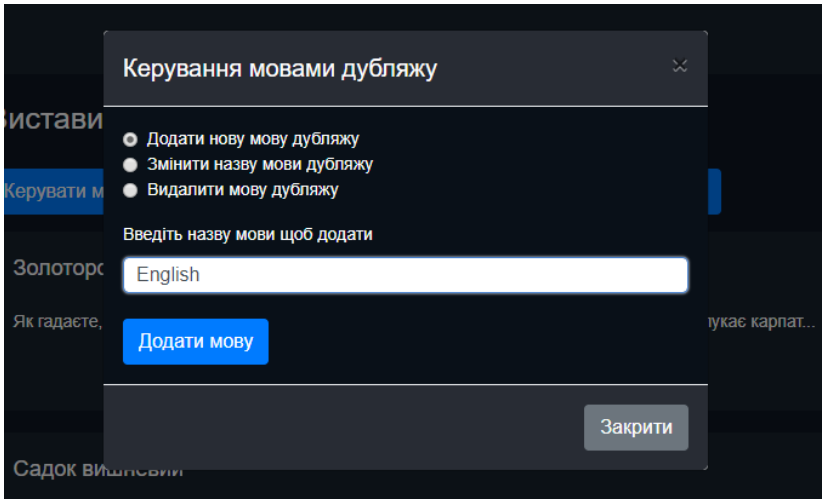


Рисунок 5.3 – Додавання мови дубляжу

Можна змінювати назву мови, для цього потрібно обрати пункт «Змінити назву мови дубляжу», обрати мову в спливаючому списку і змінити назву в текстовому полі та натиснути кнопку «Редагувати» (рисунок 5.4).

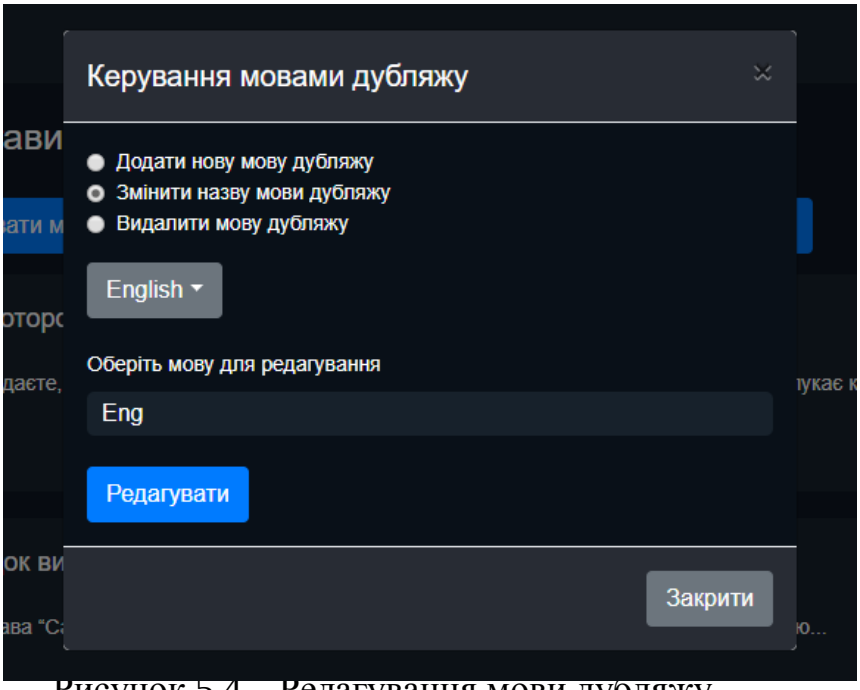


Рисунок 5.4 – Редагування мови дубляжу

Також є можливість видалити мову. **Важливо,** при видаленні мови, видаляються усі аудіо-файли цієї мови. Для видалення мови потрібно обрати пункт «Видалити мову дубляжу», обрати мову в спливаючому вікні та натиснути на кнопку «Видалити мову», після чого з’явиться вікно підтвердження, де потрібно підтвердити видалення (рисунок 5.5).

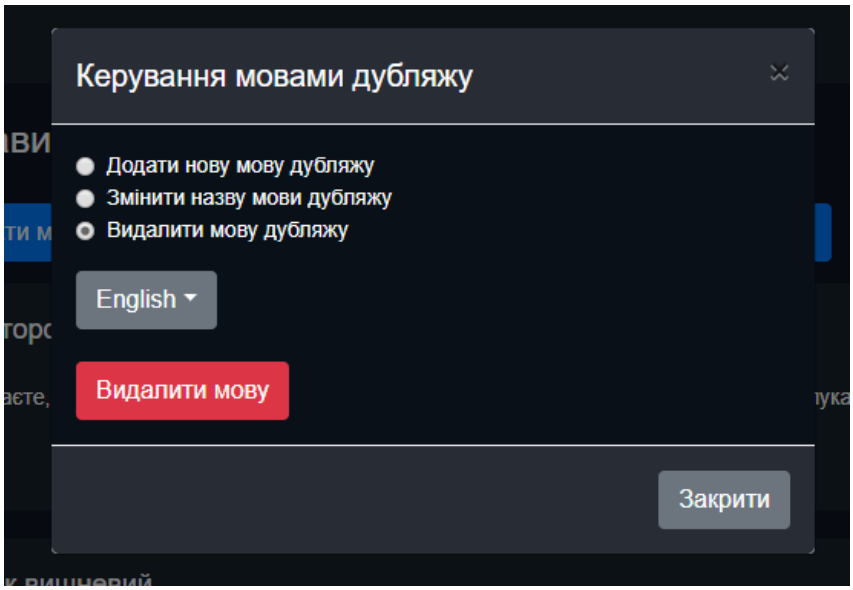


Рисунок 5.5 – Видалення мови дубляжу

Перед початком транслювання вистави, глядачу буде програватись аудіо-файл. Це потрібно для того, щоб користувач зрозумів, що він під’єднався до трансляції і все працює. Щоб змінити цей аудіо-файл потрібно на головній сторінці натиснути кнопку «Завантажити аудіо для очікування трансляції».

Тепер все готово для створення вистави. Щоб створити виставу потрібно на головній сторінці натиснути на кнопку «Додати виставу», після чого здійсниться перехід на сторінку створення вистави (рисунок 5.6).

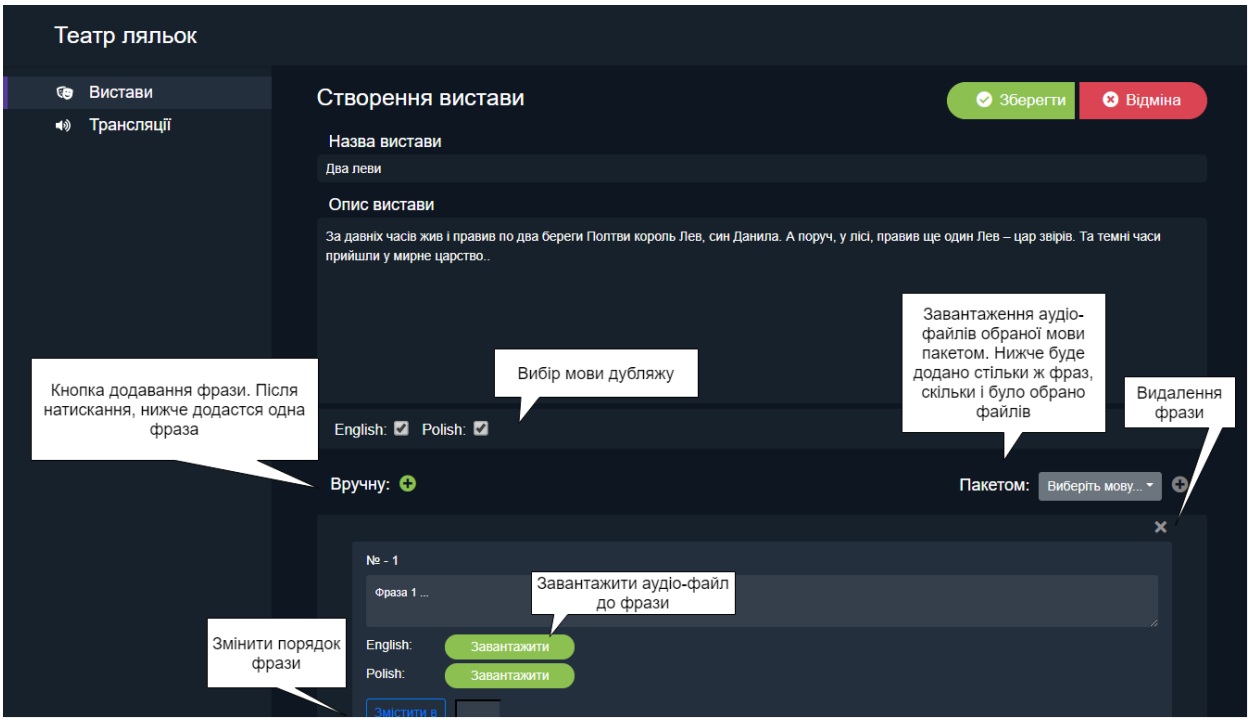


Рисунок 5.6 – Сторінка створення вистави

Спочатку потрібно ввести назву та опис вистави, обрати мови дубляжу та створити фрази. Є два способи створити фрази:

- Створювати по одній фразі (вручну);
- Створити відразу багато фраз (пакетом).

Розглянемо кожен із цих варіантів. При створюванні фрази вручну потрібно натиснути на кнопку «Вручну: ⊕», після чого нижче буде додано пусту фразу. До цієї фрази потрібно ввести її текст та завантажити аудіо-файли.

При створенні фраз пакетом потрібно обрати мову в спливаючому списку та натиснути на кнопку «Пакетом: ⊕», після чого потрібно обрати файли. До кожного файлу буде створено фразу, текстом фрази буде назва файлу без розширення, наприклад, файл – Фраза1.mp3, текст фрази – Фраза1. **Важливо**, при додаванні файлів пакетом наступної мови, назва файлу має співпадати.

Після того, як ви переконались, що виконали все вище потрібно натиснути на кнопку «Зберегти». Готово, вистава створена.

Після того, як ми створили виставу, тепер її можна транслювати слухачам. Для цього потрібно на головній сторінці біля вистави натиснути кнопку «Перейти до вистави». Буде здійснено перехід на сторінку трансляції (рисунок 5.7).

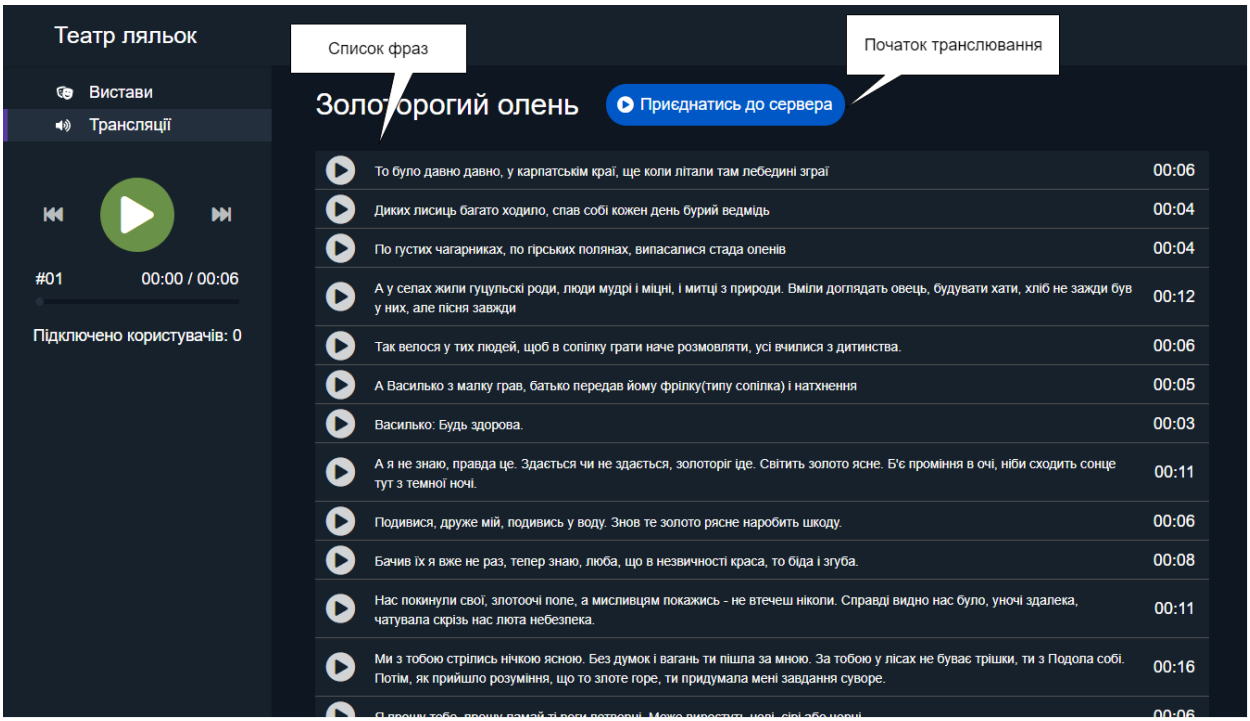


Рисунок 5.7 – Сторінка транслювання вистави

Щоб почати трансляцію потрібно натиснути на кнопку «Приєднатись до сервера». Сервер почне транслювати аудіо. Для керування можна користуватись кнопками або гарячими кнопками:

- Пробіл – play/pause;
- Ctrl + стрілка вправо – наступна фраза;

– Ctrl + стрілка вліво – попередня фраза.

Тепер розглянемо застосунок зі сторони відвідувача театру. Щоб почати роботу потрібно в браузері перейти по адресі http://localhost:5000/. Відбудеться перехід на сторінку з вибором вистав (рисунок 5.8).

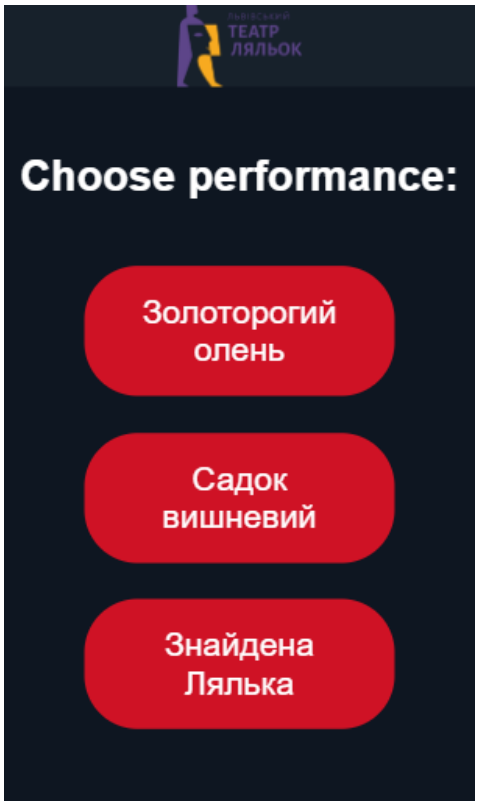


Рисунок 5.8 – Початкова сторінка вибору вистави

Далі потрібно обрати мову (рисунок 5.9).

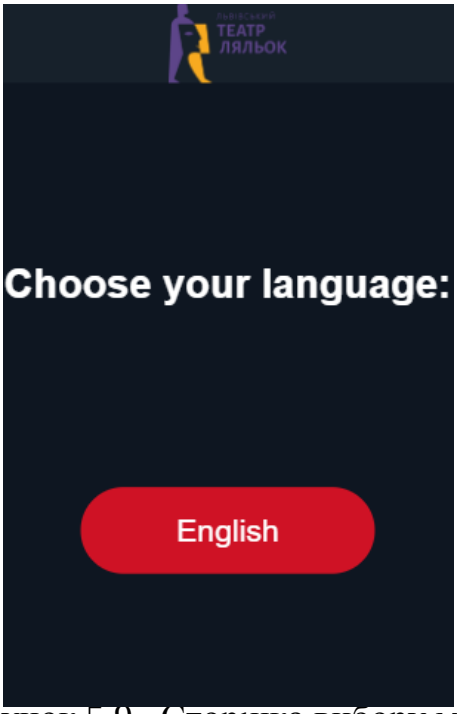


Рисунок 5.9 –Сторінка вибору мову

Після вибору мови потрібно натиснути на кнопку «Connect to stream» (рисунок 5.10). Далі потрібно просто очікувати початку вистави. Під час очікування буде грати мелодія.

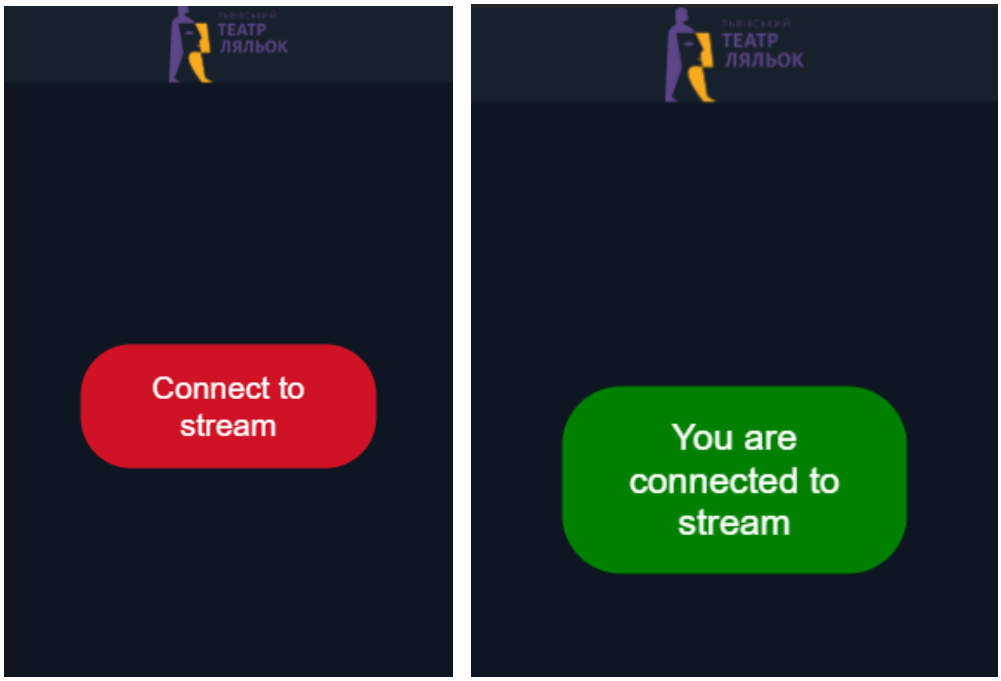


Рисунок 5.10 – Сторінка очікування трансляції

5.2 Випробування програмного продукту

В цьому підрозділі наведено опис тестів і порядок їх виконання для перевірки відповідності програмного забезпечення комплексу задач функціональним вимогам, представленим у технічному завданні на створення комплексу задач транслювання дубляжу перекладу вистави.

5.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій комплексу задач транслювання дубляжу перекладу вистави вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

В процесі тестування були перевірена уся функціональність комплексу задач (КЗ). У наступних таблицях наведений перелік випробувань основних функціональних можливостей (табл. 5.1 – 5.14).

Таблиця 5.1 – Додавання мови дубляжу

Мета тесту:	**Перевірка функції «Додавання мови дубляжу»**
Початковий стан КЗ	Відкрита «Головна» сторінка сайту
Вхідні данні:	Назва мови
Схема проведення тесту:	Натиснути кнопку «Керувати мовами дубляжу». Обрати пункт «Додати нову мову дубляжу».

Продовження таблиці 5.1

Мета тесту:	**Перевірка функції «Додавання мови дубляжу»**
	Ввести у поле «Назва мови» назву мови і натиснути на кнопку «Додати мову»
Очікуваний результат:	Додалася нова мова. Відкрита головна сторінка
Стан КЗ після проведення випробувань:	Відкрита головна сторінка

Таблиця 5.2 – Перевірка заповнення поля «Назва мови»

Мета тесту:	**Перевірка функції «Перевірка заповнення поля»**
Початковий стан КЗ	Відкрита «Головна» сторінка сайту
Вхідні данні:	Пустий рядок
Схема проведення тесту:	Натиснути кнопку «Керувати мовами дубляжу». Обрати пункт «Додати нову мову дубляжу». Ввести у поле «Назва мови» пустий рядок і натиснути на кнопку «Додати мову»
Очікуваний результат:	Вікно повідомлення про помилку вводу. Додавання мови не виконано
Стан КЗ після проведення випробувань:	Відкрите спливаюче вікно «Керування мовами дубляжу»

Таблиця 5.3 – Перевірка валідності введених даних

Мета тесту:	**Перевірка функції «Перевірка правильності введених даних»**
Початковий стан КЗ	Відкрита «Головна» сторінка сайту
Вхідні данні:	Існуюча мова дубляжу
Схема проведення тесту:	Натиснути кнопку «Керувати мовами дубляжу». Обрати пункт «Додати нову мову дубляжу». Ввести у поле «Назва мови» існуючу мову і

Продовження таблиці 5.3

Мета тесту:	**Перевірка функції «Перевірка правильності введених даних»**
	натиснути на кнопку «Додати мову»
Очікуваний результат:	Вікно повідомлення про помилку вводу. Додавання мови не виконано
Стан КЗ після проведення випробувань:	Відкрите спливаюче вікно «Керування мовами дубляжу»

Таблиця 5.4 – Редагування мови дубляжу

Мета тесту:	**Перевірка функції «Редагування мови дубляжу»**
Початковий стан КЗ:	Відкрита «Головна» сторінка сайту
Вхідні дані:	Назва мови, нова назва мови
Схема проведення тесту:	Натиснути кнопку «Керувати мовами дубляжу». Обрати пункт «Змінити назву мови дубляжу». Вибрати мову із списку і редагувати поле «Назва мови». Натиснути на кнопку «Редагувати»
Очікуваний результат:	Мову змінено. Відкрита головна сторінка сайту
Стан КЗ після проведення випробування:	Відкрита головна сторінка сайту

Таблиця 5.5 – Редагування невибраної мови дубляжу

Мета тесту:	**Перевірка функції «Редагування невибраної мови дубляжу»**
Початковий стан КЗ:	Відкрита «Головна» сторінка сайту
Вхідні дані:	Нова назва мови

Мета тесту:	**Перевірка функції «Редагування невибраної мови дубляжу»**
Схема проведення тесту:	Натиснути кнопку «Керувати мовами дубляжу». Обрати пункт «Змінити назву мови дубляжу». Редагувати поле «Назва мови». Натиснути на кнопку «Редагувати»
Очікуваний результат:	Вікно повідомлення про помилку. Редагування не виконано
Стан КЗ після проведення випробування:	Відкрите спливаюче вікно «Керування мовами дубляжу»

Таблиця 5.6 – Редагування мови дубляжу на існуючу мову

Мета тесту:	**Перевірка функції «Редагування мови дубляжу на існуючу мову»**
Початковий стан КЗ:	Відкрита «Головна» сторінка сайту
Вхідні дані:	Назва мови, існуюча назва мови
Схема проведення тесту:	Натиснути кнопку «Керувати мовами дубляжу». Обрати пункт «Змінити назву мови дубляжу». Вибрати мову із списку і в поле «Назва мови» ввести існуючу мову. Натиснути на кнопку «Редагувати»
Очікуваний результат:	Вікно повідомлення про помилку. Редагування не виконано
Стан КЗ після проведення випробування:	Відкрите спливаюче вікно «Керування мовами дубляжу»

Таблиця 5.7 – Видалення мови дубляжу

Мета тесту:	**Перевірка функції «Видалення мови дубляжу»**
Початковий стан КЗ:	Відкрита «Головна» сторінка сайту
Вхідні дані:	Назва мови
Схема проведення тесту:	Натиснути кнопку «Керувати мовами дубляжу». Обрати пункт «Видалити мови дубляжу». Вибрати мову із списку і натиснути на кнопку «Видалити мову». В попереджувальному повідомленні натиснути видалити.
Очікуваний результат:	Мову видалено. Відкрита головна сторінка сайту
Стан КЗ після проведення випробування:	Відкрита головна сторінка сайту

Таблиця 5.8 – Видалення необраної мови дубляжу

Мета тесту:	**Перевірка функції «Видалення необраної мови дубляжу»**
Початковий стан КЗ:	Відкрита «Головна» сторінка сайту
Вхідні дані:	
Схема проведення тесту:	Натиснути кнопку «Керувати мовами дубляжу». Обрати пункт «Видалити мови дубляжу». Натиснути на кнопку «Видалити мову».
Очікуваний результат:	Вікно повідомлення про помилку. Видалення не виконано
Стан КЗ після проведення випробування:	Відкрите спливаюче вікно «Керування мовами дубляжу»

Таблиця 5.9 – Видалення вистави

Мета тесту:	**Перевірка функції «Видалення вистави»**

Мета тесту:	**Перевірка функції «Видалення вистави»**
Початковий стан КЗ:	Відкрита «Головна» сторінка сайту
Вхідні дані:	Назва мови
Схема проведення тесту:	Натиснути кнопку **X**. В попереджувальному повідомленні натиснути видалити.
Очікуваний результат:	Виставу видалено. Відкрита головна сторінка сайту
Стан КЗ після проведення випробування:	Відкрита головна сторінка сайту

Таблиця 5.10 – Перехід на сторінку редагування вистави

Мета тесту:	**Перевірка функції «Редагування мови дубляжу»**
Початковий стан КЗ:	Відкрита «Головна» сторінка сайту
Вхідні дані:	
Схема проведення тесту:	Натиснути кнопку .
Очікуваний результат:	Сторінка редагування вистави
Стан КЗ після проведення випробування:	Сторінка редагування вистави

Таблиця 5.11 – Перехід на сторінку створення вистави

Мета тесту:	**Перевірка функції «Перехід на сторінку створення вистави»**
Початковий стан КЗ:	Відкрита «Головна» сторінка сайту
Вхідні дані:	
Схема проведення тесту:	Натиснути кнопку «Додати виставу» .
Очікуваний результат:	Сторінка створення вистави
Стан КЗ після проведення випробування:	Сторінка створення вистави

Продовження таблиці 5,12

Таблиця 5.12 – Перехід на сторінку транслювання вистави

Мета тесту:	**Перевірка функції «Перехід на сторінку транслювання вистави»**
Початковий стан КЗ:	Відкрита «Головна» сторінка сайту
Вхідні дані:	
Схема проведення тесту:	Натиснути кнопку «Перейти до вистави» .
Очікуваний результат:	Сторінка транслювання вистави
Стан КЗ після проведення випробування:	Сторінка транслювання вистави

Таблиця 5.13 – Початок транслювання

Мета тесту:	**Перевірка функції «Початок транслювання»**
Початковий стан КЗ:	Відкрита «Головна» сторінка сайту
Вхідні дані:	
Схема проведення тесту:	Натиснути кнопку «Перейти до вистави». Натиснути кнопку «Приєднатись до сервера»
Очікуваний результат:	Сторінка транслювання вистави
Стан КЗ після проведення випробування:	Сторінка транслювання вистави

Таблиця 5.14 – Завантаження аудіо для очікування вистави

Мета тесту:	**Перевірка функції «Початок транслювання»**
Початковий стан КЗ:	Відкрита «Головна» сторінка сайту
Вхідні дані:	

Мета тесту:	**Перевірка функції «Початок транслювання»**
Схема проведення тесту:	Натиснути кнопку «Завантажити аудіо для очікування вистави». Обрати файл і натиснути «Ок»
Очікуваний результат:	Завантажено файл очікування в файлову систему
Стан КЗ після проведення випробування:	Головна сторінка

Висновок до розділу

В даному розділі було створено керівництво користувача з інструкцією по користуванню. Було протестовано програмний продукт.

ЗАГАЛЬНІ ВИСНОВКИ

У ході виконання дипломного проекту було детально розглянуто предметну область. Проведено аналіз існуючих аналогів та визначено плюси та мінуси. Були виділені основні ключові етапи, властиві процесу, і взаємозв'язку між ними.

На основі отриманої інформації було визначено вхідні та вихідні дані. На їх основі розроблено структуру бази даних та проведено аналіз сутностей. Після проведення аналіз баз даних було обрано базу даних SQLite.

Для розробки Back-end частини програмного забезпечення, було обрано мову програмування C# та фреймворк .NET Core. Для розробки була використана програмна платформа Microsoft Visual Studio 2017 Community.

Для розробки Front-end частини, було обрано мову TypeScript, HTML, CSS та фреймворк ReactJS. Для розробки використовувався редактор коду Visual Code та фреймворки WebPack та NodeJS Package Manager.

Створено детальну інструкцію по користуванню продуктом для користувача описана методика проведення випробувань, яка показує можливість введення програми в експлуатацію

ПЕРЕЛІК ПОСИЛАНЬ

1. Metanit. ASP.NET WebAPI [Електронний ресурс]. – Режим доступу: https://metanit.com/sharp/aspnet_webapi/1.1.php - Дата доступу: 20.02.2019.
2. React Tutorial [Електронний ресурс]. – Режим доступу: https://reactjs.org/tutorial/tutorial.html - Дата доступу: 08.02.2019.
3. Microsoft Docs. SignalR – Режим доступу: https://docs.microsoft.com/ru-ru/aspnet/core/tutorials/signalr?view=aspnetcore-2.2&tabs=visual-studio - Дата доступу: 20.01.2019.
4. Professor Web [Електронний ресурс]. – Режим доступу: http://professorweb.ru - Дата доступу: 22.03.2019.

Додаток А

Тексти програмного коду

Система підтримки синхронного дубляжу вистав

(Найменування програми (документа))

DVD-R

(Вид носія даних)

22 арк, 25 Мб

(Обсяг програми (документа) , арк.,) Мб)

Київ – 2019 року

Лістинг коду

Web/ApiControllers/AudioController.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.DTOs;

namespace SoftServe.ITAcademy.BackendDubbingProject.Web.ApiControllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AudioController : ControllerBase
    {
        private readonly IAdministrationService _administrationMicroservice;

        public AudioController(IAdministrationService administrationMicroservice)
        {
            _administrationMicroservice = administrationMicroservice;
        }

        /// <summary>Controller method for getting a list of all audios.</summary>
        /// <returns>List of all audios.</returns>
        /// <response code="200">Is returned when the list has at least one audio.</response>
        /// <response code="404">Is returned when the list of audios is empty.</response>
        [HttpGet]
        public async Task<ActionResult<List<AudioDTO>>> GetAll()
        {
            var listOfAudiosDTOs = await _administrationMicroservice.GetAllAudiosAsync();

            return Ok(listOfAudiosDTOs);
        }

        /// <summary>Controller method for getting a audio by id.</summary>
        /// <param name="id">Id of audio that need to receive.</param>
        /// <returns>The audio with the following id.</returns>
        /// <response code="200">Is returned when audio does exist.</response>
        /// <response code="404">Is returned when audio with such Id doesn't exist.</response>
        [HttpGet("{id}")]
        public async Task<ActionResult<AudioDTO>> GetById(int id)
        {
            var audioDTO = await _administrationMicroservice.GetAudioByIdAsync(id);

            if (audioDTO == null)
                return NotFound();

            return Ok(audioDTO);
        }

        /// <summary>Controller method for creating new audio.</summary>
        /// <param name="audioDTO">Audio model which needed to create.</param>
        /// <returns>Status code and audio.</returns>
        /// <response code="201">Is returned when audio is successfully created.</response>
        /// <response code="400">Is returned when invalid data is passed.</response>
        /// <response code="409">Is returned when audio with such parameters already exists.</response>
        [HttpPost]
        public async Task<ActionResult<AudioDTO>> Create(AudioDTO audioDTO)
        {
            await _administrationMicroservice.CreateAudioAsync(audioDTO);

            return CreatedAtAction(nameof(GetById), new {id = audioDTO.Id}, audioDTO);
        }

        /// <summary>Controller method for uploading a file to server and saving it to a local storage.</summary>
        /// <returns>Status code, URL of audio file and audio model.</returns>
        /// <param name="audioFileDTO">Audio file which needed to create.</param>
        /// <response code="201">Is returned when audio is successfully uploaded.</response>
        /// <response code="400">Is returned when invalid data is passed.</response>
        [HttpPost("upload")]
        public async Task<ActionResult<AudioFileDTO>> Upload([FromForm] AudioFileDTO audioFileDTO)
        {
            await _administrationMicroservice.UploadAudioAsync(audioFileDTO);

            var urlOfAudioFile = HttpContext.Request.Host.Value + "/audio/" + audioFileDTO.File.FileName;

            return Created(urlOfAudioFile, audioFileDTO);
        }

        /// <summary>Controller method for updating an already existing audio with following id.</summary>
        /// <param name="id">Id of the audio that is needed to be updated.</param>
        /// <param name="audioDTO">The audio model to which is needed to be updated existing audio.</param>
        /// <returns>Status code and optionally exception message.</returns>
        /// <response code="204">Is returned when speech is successfully updated.</response>
        /// <response code="400">Is returned when speech with or invalid data is passed.</response>
        /// <response code="404">Is returned when speech with such Id is not founded</response>
        [HttpPut("{id}")]
```

```
        public async Task<ActionResult> Update(int id, AudioDTO audioDTO)
        {
            if (audioDTO.Id != id)
                BadRequest();

            await _administrationMicroservice.UpdateAudioAsync(id, audioDTO);

            return NoContent();
        }

        /// <summary>
        /// Unload audio from server
        /// </summary>
        /// <param name="files"></param>
        /// <returns>Delete</returns>
        [HttpDelete("unload")]
        public ActionResult Delete([FromQuery] string[] files)
        {
            _administrationMicroservice.DeleteAudioFiles(files);

            return NoContent();
        }

        /// <summary>
        /// Delete info about audio from DB
        /// </summary>
        /// <param name="id"></param>
        /// <returns>Delete</returns>
        [HttpDelete("{id}")]
        public ActionResult Delete(int id)
        {
            _administrationMicroservice.DeleteFileAsync(id);

            return NoContent();
        }

        /// <summary>Controller method for uploading a file to server and saving it to a local storage.</summary>
        /// <returns>Status code, URL of audio file and audio model.</returns>
        /// <param name="audioFileDTO">Audio file which needed to create.</param>
        /// <response code="201">Is returned when audio is successfully uploaded.</response>
        /// <response code="400">Is returned when invalid data is passed.</response>
        [HttpPost("upload/waiting")]
        public async Task<ActionResult<AudioFileDTO>> UploadWaiting([FromForm] AudioFileDTO audioFileDTO)
        {
            await _administrationMicroservice.UploadWaitingAudioAsync(audioFileDTO);

            var urlOfAudioFile = HttpContext.Request.Host.Value + "/audio/" + audioFileDTO.File.FileName;

            return Created(urlOfAudioFile, audioFileDTO);
        }
    }
}
```

Web/ApiControllers/LanguageController.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.DTOs;

namespace SoftServe.ITAcademy.BackendDubbingProject.Web.ApiControllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class LanguageController : ControllerBase
    {
        private readonly IAdministrationService _administrationMicroservice;

        public LanguageController(IAdministrationService administrationMicroservice)
        {
            _administrationMicroservice = administrationMicroservice;
        }

        /// <summary>Controller method for getting a list of all languages.</summary>
        /// <returns>List of all languages.</returns>
        /// <response code="200">Is returned when the list has at least one language.</response>
        /// <response code="404">Is returned when the list of languages is empty.</response>
        [HttpGet]
        public async Task<ActionResult<List<LanguageDTO>>> GetAll()
        {
            var listOfLanguageDTOs = await _administrationMicroservice.GetAllLanguagesAsync();

            return Ok(listOfLanguageDTOs);
        }

        /// <summary>Controller method for getting a language by id.</summary>
        /// <param name="id">Id of language that need to receive.</param>
```

```
        /// <returns>The language with the following id.</returns>
        /// <response code="200">Is returned when language does exist.</response>
        /// <response code="404">Is returned when language with such Id doesn't exist.</response>
        [HttpGet("{id}")]
        public async Task<ActionResult<LanguageDTO>> GetById(int id)
        {
            var languageDTO = await _administrationMicroservice.GetLanguageByIdAsync(id);

            if (languageDTO == null)
                return NotFound();

            return Ok(languageDTO);
        }

        /// <summary>Controller method for creating new language.</summary>
        /// <param name="languageDTO">Language model which needed to create.</param>
        /// <returns>Status code and Language.</returns>
        /// <response code="201">Is returned when language is successfully created.</response>
        /// <response code="400">Is returned when invalid data is passed.</response>
        /// <response code="409">Is returned when language with such parameters already exists.</response>
        [HttpPost]
        public async Task<ActionResult<LanguageDTO>> Create(LanguageDTO languageDTO)
        {
            await _administrationMicroservice.CreateLanguageAsync(languageDTO);

            return CreatedAtAction(nameof(GetById), new {id = languageDTO.Id}, languageDTO);
        }

        /// <summary>Controller method for updating an already existing language with following id.</summary>
        /// <param name="id">Id of the language that is needed to be updated.</param>
        /// <param name="languageDTO">The language model that is needed to be created.</param>
        /// <returns>Status code and optionally exception message.</returns>
        /// <response code="204">Is returned when language is successfully updated.</response>
        /// <response code="400">Is returned when language with or invalid data is passed.</response>
        /// <response code="404">Is returned when language with such Id is not founded</response>
        [HttpPut("{id}")]
        public async Task<ActionResult> Update(int id, LanguageDTO languageDTO)
        {
            if (languageDTO.Id != id)
                BadRequest();

            await _administrationMicroservice.UpdateLanguageAsync(id, languageDTO);

            return NoContent();
        }

        /// <summary>Controller method for deleting an already existing language with following id.</summary>
        /// <param name="id">Id of the language that needed to delete.</param>
        /// <returns>Status code</returns>
        /// <response code="204">Is returned when language is successfully deleted.</response>
        [HttpDelete("{id}")]
        public async Task<ActionResult> Delete(int id)
        {
            await _administrationMicroservice.DeleteLanguageAsync(id);

            return NoContent();
        }
    }
}
```

Web/ApiControllers/PerformanceController.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.DTOs;

namespace SoftServe.ITAcademy.BackendDubbingProject.Web.ApiControllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class PerformanceController : ControllerBase
    {
        private readonly IAdministrationService _administrationMicroservice;

        public PerformanceController(IAdministrationService administrationMicroservice)
        {
            _administrationMicroservice = administrationMicroservice;
        }

        /// <summary>Controller method for getting a list of all performances.</summary>
        /// <returns>List of all performances.</returns>
        /// <response code="200">Is returned when the list has at least one performance.</response>
        /// <response code="404">Is returned when the list of performances is empty.</response>
        [HttpGet]
        public async Task<ActionResult<List<PerformanceDTO>>> GetAll()
        {
            var listOfPerformanceDTOs = await _administrationMicroservice.GetAllPerformancesAsync();
```

```
    return Ok(listOfPerformanceDTOs);
}

/// <summary>Controller method for getting a performance by id.</summary>
/// <param name="id">Id of performance that need to receive.</param>
/// <returns>The performance with the following id.</returns>
/// <response code="200">Is returned when performance does exist.</response>
/// <response code="404">Is returned when performance with such Id doesn't exist.</response>
[HttpGet("{id}")]
public async Task<ActionResult<PerformanceDTO>> GetById(int id)
{
    var performanceDTO = await _administrationMicroservice.GetPerformanceByIdAsync(id);

    if (performanceDTO == null)
        return NotFound();

    return Ok(performanceDTO);
}

/// <summary>Controller method for creating new performance.</summary>
/// <param name="performanceDTO">Performance model which needed to create.</param>
/// <returns>Status code and performance.</returns>
/// <response code="201">Is returned when performance is successfully created.</response>
/// <response code="400">Is returned when invalid data is passed.</response>
/// <response code="409">Is returned when performance with such parameters already exists.</response>
[HttpPost]
public async Task<ActionResult<PerformanceDTO>> Create(PerformanceDTO performanceDTO)
{
    await _administrationMicroservice.CreatePerformanceAsync(performanceDTO);

    return CreatedAtAction(nameof(GetById), new {id = performanceDTO.Id}, performanceDTO);
}

/// <summary>Controller method for updating an already existing performance with following id.</summary>
/// <param name="id">Id of the performance that is needed to be updated.</param>
/// <param name="performanceDTO">The performance model that is needed to be created.</param>
/// <returns>Status code and optionally exception message.</returns>
/// <response code="204">Is returned when performance is successfully updated.</response>
/// <response code="400">Is returned when performance with or invalid data is passed.</response>
/// <response code="404">Is returned when performance with such Id is not founded</response>
[HttpPut("{id}")]
public async Task<ActionResult> Update(int id, PerformanceDTO performanceDTO)
{
    if (performanceDTO.Id != id)
        BadRequest();

    await _administrationMicroservice.UpdatePerformanceAsync(id, performanceDTO);

    return NoContent();
}

/// <summary>Controller method for deleting an already existing performance with following id.</summary>
/// <param name="id">Id of the performance that needed to delete.</param>
/// <returns>Status code</returns>
/// <response code="204">Is returned when performance is successfully deleted.</response>
[HttpDelete("{id}")]
public async Task<ActionResult> Delete(int id)
{
    await _administrationMicroservice.DeletePerformanceAsync(id);

    return NoContent();
}

/// <summary>Controller method for getting a speeches by id of performance.</summary>
/// <param name="id">Id of performance which speeches that need to receive.</param>
/// <returns>List of a speeches.</returns>
/// <response code="200">Is returned when speeches does exist.</response>
/// <response code="400">Is returned when performance with such Id doesn't exist.</response>
/// <response code="404">Is returned when speeches doesn't exist.</response>
[HttpGet("{id}/speeches")]
public async Task<ActionResult<List<SpeechDTO>>> GetByIdWithChildren(int id)
{
    var listOfSpeechDTOs = await _administrationMicroservice.GetSpeechesByPerformanceIdAsync(id);

    if (listOfSpeechDTOs == null)
        return BadRequest($"Performance with Id: {id} doesn't exist!");

    return Ok(listOfSpeechDTOs);
}

/// <summary>dd</summary>
/// <param name="id"></param>
/// <returns>ff</returns>
/// <response code="200"></response>
/// <response code="400"></response>
/// <response code="404"></response>
[HttpGet("{id}/languages")]
```

```
        public async Task<ActionResult<List<LanguageDTO>>> GetByIdLanguages(int id)
        {
            var listOfLanguagesDTOs = await _administrationMicroservice.GetLanguagesByPerformanceIdAsync(id);

            return Ok(listOfLanguagesDTOs);
        }
    }
}
```

Web/ApiControllers/SpeechController.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.DTOs;

namespace SoftServe.ITAcademy.BackendDubbingProject.Web.ApiControllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class SpeechController : ControllerBase
    {
        private readonly IAdministrationService _administrationMicroservice;

        public SpeechController(IAdministrationService administrationMicroservice)
        {
            _administrationMicroservice = administrationMicroservice;
        }

        /// <summary>Controller method for getting a list of all speeches.</summary>
        /// <returns>List of all speeches.</returns>
        /// <response code="200">Is returned when the list has at least one speech.</response>
        /// <response code="404">Is returned when the list of speeches is empty.</response>
        [HttpGet]
        public async Task<ActionResult<List<SpeechDTO>>> GetAll()
        {
            var listOfSpeechesDTOs = await _administrationMicroservice.GetAllSpeechesAsync();

            return Ok(listOfSpeechesDTOs);
        }

        /// <summary>Controller method for getting a speech by id.</summary>
        /// <param name="id">Id of speech that need to receive.</param>
        /// <returns>The speech with the following id.</returns>
        /// <response code="200">Is returned when speech does exist.</response>
        /// <response code="404">Is returned when speech with such Id doesn't exist.</response>
        [HttpGet("{id}")]
        public async Task<ActionResult<SpeechDTO>> GetById(int id)
        {
            var speechDTO = await _administrationMicroservice.GetSpeechByIdAsync(id);

            if (speechDTO == null)
                return NotFound();

            return Ok(speechDTO);
        }

        /// <summary>Controller method for creating new speech.</summary>
        /// <param name="speechDTO">Speech model which needed to create.</param>
        /// <returns>Status code and speech.</returns>
        /// <response code="201">Is returned when speech is successfully created.</response>
        /// <response code="400">Is returned when invalid data is passed.</response>
        /// <response code="409">Is returned when speech with such parameters already exists.</response>
        [HttpPost]
        public async Task<ActionResult<SpeechDTO>> Create(SpeechDTO speechDTO)
        {
            await _administrationMicroservice.CreateSpeechAsync(speechDTO);

            return CreatedAtAction(nameof(GetById), new {id = speechDTO.Id}, speechDTO);
        }

        /// <summary>Controller method for updating an already existing speech with following id.</summary>
        /// <param name="id">Id of the speech that is needed to be updated.</param>
        /// <param name="speechDTO">The speech model that is needed to be updated.</param>
        /// <returns>Status code and optionally exception message.</returns>
        /// <response code="204">Is returned when speech is successfully updated.</response>
        /// <response code="400">Is returned when speech with or invalid data is passed.</response>
        /// <response code="404">Is returned when speech with such Id is not founded</response>
        [HttpPut("{id}")]
        public async Task<ActionResult> Update(int id, SpeechDTO speechDTO)
        {
            if (speechDTO.Id != id)
                BadRequest();

            await _administrationMicroservice.UpdateSpeechAsync(id, speechDTO);

            return NoContent();
        }
```

```
        /// <summary>Controller method for deleting an already existing speech with following id.</summary>
        /// <param name="id">Id of the speech that needed to delete.</param>
        /// <returns>Status code</returns>
        /// <response code="204">Is returned when speech is successfully deleted.</response>
        [HttpDelete("{id}")]
        public async Task<ActionResult> Delete(int id)
        {
            await _administrationMicroservice.DeleteSpeechAsync(id);

            return NoContent();
        }

        /// <summary>Controller method for getting a speeches by id of performance.</summary>
        /// <param name="id">Id of performance which speeches that need to receive.</param>
        /// <returns>List of a speeches.</returns>
        /// <response code="200">Is returned when speeches does exist.</response>
        /// <response code="400">Is returned when performance with such Id doesn't exist.</response>
        /// <response code="404">Is returned when speeches doesn't exist.</response>
        [HttpGet("{id}/audios")]
        public async Task<ActionResult<List<AudioDTO>>> GetByIdWithChildren(int id)
        {
            var listOfAudioDTOs = await _administrationMicroservice.GetAudiosBySpeechIdAsync(id);

            if (listOfAudioDTOs == null)
                return BadRequest($"Speech with Id: {id} doesn't exist!");

            return Ok(listOfAudioDTOs);
        }
    }
}
```

Web/Program.cs

```
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.DependencyInjection;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Configuration;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Infrastructure.Configuration;

namespace SoftServe.ITAcademy.BackendDubbingProject.Web
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateWebHostBuilder(args).Build().Run();
        }

        public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .ConfigureServices(services =>
                    services.AddTransient<IAdministrationServiceCollection, AdministrationServiceCollection>())
                .ConfigureServices(services =>
                    services.AddTransient<IInfrastructureServiceCollection, InfrastructureServiceCollection>())
                .UseStartup<Startup>();
    }
}
```

Web/Startup.cs

```
using System;
using System.IO;
using System.Reflection;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.FileProviders;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Configuration;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Infrastructure.Configuration;
using SoftServe.ITAcademy.BackendDubbingProject.Streaming.Core.Hubs;
using Swashbuckle.AspNetCore.Swagger;

namespace SoftServe.ITAcademy.BackendDubbingProject.Web
{
    public class Startup
    {
        private const string CorsName = "AllowAllOrigins";
        private readonly IAdministrationServiceCollection _administrationServiceCollection;
        private readonly IInfrastructureServiceCollection _infrastructureServiceCollection;

        public Startup(
            IConfiguration configuration,
            IAdministrationServiceCollection administrationServiceCollection,
            IInfrastructureServiceCollection infrastructureServiceCollection)
        {
            Configuration = configuration;
            _administrationServiceCollection = administrationServiceCollection;
```

```
            _infrastructureServiceCollection = infrastructureServiceCollection;
        }

        public IConfiguration Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddCors(options =>
            {
                options.AddPolicy(
                    CorsName,
                    builder =>
                    {
                        builder
                            .SetIsOriginAllowed(host => true)
                            .AllowAnyMethod()
                            .AllowAnyHeader()
                            .AllowCredentials();
                    });
            });

            AddAdministrationServices(services);

            AddInfrastructureServices(services);

            services.AddSignalR();

            services.AddSwaggerGen(c =>
            {
                c.SwaggerDoc("v1", new Info {Title = "APIs", Version = "v1"});
                var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
                var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);
                c.IncludeXmlComments(xmlPath);
            });

            services
                .AddMvc()
                .SetCompatibilityVersion(CompatibilityVersion.Version_2_1)
                .AddJsonOptions(options =>
                    options.SerializerSettings.ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.Ignore);
        }

        private void AddAdministrationServices(IServiceCollection services)
        {
            _administrationServiceCollection.RegisterDependencies(services);
        }

        private void AddInfrastructureServices(IServiceCollection services)
        {
            _infrastructureServiceCollection.RegisterDependencies(services);
        }

        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseHsts();
            }

            app.UseCors(CorsName);

            app.UseSwagger();

            app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "APIs V1"));

            app.UseFileServer(new FileServerOptions
            {
                FileProvider = new PhysicalFileProvider(
                    Path.Combine(Directory.GetCurrentDirectory(), "AudioFiles")),
                RequestPath = "/audio",
                EnableDirectoryBrowsing = true
            });

            app.UseSignalR(options => options.MapHub<StreamHub>("/StreamHub"));

            app.UseDefaultFiles();

            app.UseStaticFiles();

            app.UseMvc();
        }
    }
}
```

Streaming/Hubs/StreamHub.cs

```
using System;
using System.Security.Principal;
using System.Threading.Tasks;
using Microsoft.AspNetCore.SignalR;

namespace SoftServe.ITAcademy.BackendDubbingProject.Streaming.Core.Hubs
{
    internal class StreamHub : Hub
    {
        private static int _count;
        private static bool _needWait;
        private string _adminId;

        public override async Task OnConnectedAsync()
        {
            _count++;

            await base.OnConnectedAsync();

            var numberOfConnections = (_count - 1).ToString();

            await Clients.User(_adminId).SendAsync("updateCount", numberOfConnections);

            if (_needWait)
                await Clients.Caller.SendAsync("ReceiveMessage", "Start");
        }

        public override async Task OnDisconnectedAsync(Exception exception)
        {
            _count--;

            await base.OnDisconnectedAsync(exception);

            var numberOfConnections = (_count - 1).ToString();

            await Clients.User(_adminId).SendAsync("updateCount", numberOfConnections);
        }

        public async Task SendMessage(string message)
        {
            switch (message)
            {
                case "Start":
                    _adminId = Context.ConnectionId;
                    _needWait = true;
                    break;
                case "End":
                    _adminId = null;
                    _needWait = false;
                    break;
                default:
                    _needWait = false;
                    break;
            }

            await Clients.Others.SendAsync("ReceiveMessage", message);
        }
    }
}
```

Streaming/Hubs/InternalVisibleTo.cs

```
using System.Runtime.CompilerServices;

[assembly: InternalsVisibleTo("Web")]
[assembly: InternalsVisibleTo("UnitTests")]
```

Administration/Core/DTOs/AudioDTO.cs

```
using System.ComponentModel.DataAnnotations;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.DTOs
{
    public class AudioDTO
    {
        public int Id { get; set; }

        [Required]
        [StringLength(32)]
        public string FileName { get; set; }

        [Required]
        [StringLength(256)]
        public string OriginalText { get; set; }

        [Required]
        public int SpeechId { get; set; }

        [Required]
```

```
        public int LanguageId { get; set; }
    }
}
```

Administration/Core/DTOs/AudioFileDTO.cs

```
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.Http;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.DTOs
{
    public class AudioFileDTO
    {
        public int Id { get; set; }

        [Required]
        public IFormFile File { get; set; }
    }
}
```

Administration/Core/DTOs/LanguageDTO.cs

```
using System.ComponentModel.DataAnnotations;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.DTOs
{
    public class LanguageDTO
    {
        public int Id { get; set; }

        [Required]
        [StringLength(32)]
        public string Name { get; set; }
    }
}
```

Administration/Core/DTOs/PerformanceDTO.cs

```
using System.ComponentModel.DataAnnotations;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.DTOs
{
    public class PerformanceDTO
    {
        public int Id { get; set; }

        [Required]
        [StringLength(64)]
        public string Title { get; set; }

        [Required]
        [StringLength(256)]
        public string Description { get; set; }
    }
}
```

Administration/Core/DTOs/SpeechDTO.cs

```
using System.ComponentModel.DataAnnotations;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.DTOs
{
    public class SpeechDTO
    {
        public int Id { get; set; }

        [Required]
        public int Order { get; set; }

        [Required]
        [StringLength(256)]
        public string Text { get; set; }

        public int Duration { get; set; }

        [Required]
        public int PerformanceId { get; set; }
    }
}
```

Administration/Core/Entities/Audio.cs

```
using System.ComponentModel.DataAnnotations.Schema;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities
{
    internal class Audio : BaseEntity
    {
        public string FileName { get; set; }

        public string OriginalText { get; set; }

        public int Duration { get; set; }
```

```
        public int SpeechId { get; set; }

        public Speech Speech { get; set; }

        public int LanguageId { get; set; }

        public Language Language { get; set; }

        [NotMapped]
        public byte[] AudioFile { get; set; }
    }
}
```

Administration/Core/Entities/BaseEntity.cs

```
namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities
{
    internal abstract class BaseEntity
    {
        public int Id { get; set; }
    }
}
```

Administration/Core/Entities/Language.cs

```
using System.Collections.Generic;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities
{
    internal class Language : BaseEntity
    {
        public string Name { get; set; }

        public ICollection<Audio> Audios { get; set; }
    }
}
```

Administration/Core/Entities/Performance.cs

```
using System.Collections.Generic;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities
{
    internal class Performance : BaseEntity
    {
        public string Title { get; set; }

        public string Description { get; set; }

        public ICollection<Speech> Speeches { get; set; }
    }
}
```

Administration/Core/Entities/Speech.cs

```
using System.Collections.Generic;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities
{
    internal class Speech : BaseEntity
    {
        public int Order { get; set; }

        public string Text { get; set; }

        public int Duration { get; set; }

        public int PerformanceId { get; set; }

        public Performance Performance { get; set; }

        public ICollection<Audio> Audios { get; set; }
    }
}
```

Administration/Core/Intrefaces/IAudioService.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.DTOs;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Interfaces
{
    internal interface IAudioService
    {
        Task<List<Audio>> GetAllAsync();

        Task<Audio> GetByIdAsync(int id);

        Task CreateAsync(Audio entity);

        Task UploadAsync(Audio audio, AudioFileDTO audioFileDTO);
```

```
        Task UpdateAsync(int id, Audio newEntity);

        Task DeleteAsync(int id);

        Task DeleteFileAsync(int id);

        void DeleteAudioFiles(IEnumerable<string> namesList);
    }
}
```

Administration/Core/Intrefaces/IFileSystemRepository.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Interfaces
{
    internal interface IFileSystemRepository
    {
        Task WriteToFileSystemAsync(Audio audio, string path);

        void Delete(string folderPath, IEnumerable<string> names);
    }
}
```

Administration/Core/Intrefaces/ILanuageService.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Interfaces
{
    internal interface ILanguageService
    {
        Task<List<Language>> GetAllAsync();

        Task<Language> GetByIdAsync(int id);

        Task CreateAsync(Language entity);

        Task UpdateAsync(int id, Language newEntity);

        Task DeleteAsync(int id);
    }
}
```

Administration/Core/Intrefaces/IPerformanceService.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Interfaces
{
    internal interface IPerformanceService
    {
        Task<Performance> GetByIdAsync(int id);

        Task<List<Speech>> GetChildrenByIdAsync(int id);

        Task<List<Performance>> GetAllAsync();

        Task CreateAsync(Performance entity);

        Task UpdateAsync(int id, Performance newEntity);

        Task DeleteAsync(int id);

        Task<List<Language>> GetLanguagesAsync(int id);
    }
}
```

Administration/Core/Intrefaces/IRepository.cs

```
using System;
using System.Collections.Generic;
using System.Linq.Expressions;
using System.Threading.Tasks;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Interfaces
{
    internal interface IRepository<T>
        where T : BaseEntity
    {
        Task<T> GetByIdAsync(int id);

        Task<T> GetByIdWithChildrenAsync(int id, string childrenName);

        Task<List<T>> ListAllAsync();
```

```
        Task AddAsync(T entity);

        Task UpdateAsync(T oldEntity, T newEntity);

        Task UpdateFieldAsync(T entity, string fieldName);

        Task DeleteAsync(T entity);
    }
}
```

Administration/Core/Intrefaces/ISpeechService.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Interfaces
{
    internal interface ISpeechService
    {
        Task<List<Speech>> GetAllAsync();

        Task<Speech> GetByIdAsync(int id);

        Task<List<Audio>> GetChildrenByIdAsync(int id);

        Task CreateAsync(Speech entity);

        Task UpdateAsync(int id, Speech newEntity);

        Task DeleteAsync(int id);
    }
}
```

Administration/Core/Mapping/MappingProfile.cs

```
using AutoMapper;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.DTOs;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Mapping
{
    internal class MappingProfile : Profile
    {
        public MappingProfile()
        {
            CreateMap<Performance, PerformanceDTO>();
            CreateMap<PerformanceDTO, Performance>();

            CreateMap<Speech, SpeechDTO>();
            CreateMap<SpeechDTO, Speech>();

            CreateMap<Language, LanguageDTO>();
            CreateMap<LanguageDTO, Language>();

            CreateMap<Audio, AudioFileDTO>();
            CreateMap<AudioFileDTO, Audio>()
                .ForMember("FileName", cnf => cnf.MapFrom(m => m.File.FileName));

            CreateMap<Audio, AudioDTO>();
            CreateMap<AudioDTO, Audio>();
        }
    }
}
```

Administration/Core/Services/AudioService.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.DTOs;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Interfaces;
using File = System.IO.File;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Services
{
    internal class AudioService : GenericService<Audio>, IAudioService
    {
        private readonly IFileSystemRepository _fileSystemRepository;

        private readonly IRepository<Speech> _speechRepository;

        private readonly string _audioFilesFolderPath = Path.GetFullPath("../Web/AudioFiles/");

        public AudioService(
            IRepository<Audio> repository,
            IRepository<Speech> speechRepository,
            IFileSystemRepository fileSystemRepository)
```

```
    : base(repository)
{
    _speechRepository = speechRepository;
    _fileSystemRepository = fileSystemRepository;
}

public override async Task CreateAsync(Audio entity)
{
    var audio = await ChangeName(entity);

    audio = await ChangeDuration(entity);

    entity.Id = default(int);

    await Repository.AddAsync(audio);
}

public async Task UploadAsync(Audio audio, AudioFileDTO audioFileDTO)
{
    using (var memStream = new MemoryStream())
    {
        audioFileDTO.File.CopyTo(memStream);

        audio.AudioFile = memStream.ToArray();

        if (audio.FileName != "waiting.mp3")
        {
            audio.FileName = audioFileDTO.File.FileName;
        }
    }

    var path = Path.Combine(_audioFilesFolderPath, audio.FileName);

    await _fileSystemRepository.WriteToFileSystemAsync(audio, path);
}

public override async Task UpdateAsync(int id, Audio newEntity)
{
    var oldEntity = await Repository.GetByIdAsync(id);

    if (oldEntity == null)
        throw new Exception($"{typeof(Audio)} entity with ID: {id} doesn't exist.");

    newEntity = await ChangeDuration(newEntity);

    if (oldEntity.FileName != newEntity.FileName)
    {
        var fileToRemovePath = Path.Combine(_audioFilesFolderPath, oldEntity.FileName);
        File.Delete(fileToRemovePath);

        newEntity = await ChangeName(newEntity);

        await Repository.UpdateAsync(oldEntity, newEntity);
    }
}

public override async Task DeleteAsync(int id)
{
    var speech = await _speechRepository.GetByIdWithChildrenAsync(id, "Audios");

    var namesList = speech.Audios.Select(audio => audio.FileName).AsEnumerable();

    DeleteAudioFiles(namesList);
}

public async Task DeleteFileAsync(int id)
{
    var audio = await Repository.GetByIdAsync(id);

    await Repository.DeleteAsync(audio);
}

public void DeleteAudioFiles(IEnumerable<string> namesList)
{
    _fileSystemRepository.Delete(_audioFilesFolderPath, namesList);
}

private async Task<Audio> ChangeName(Audio entity)
{
    var speech = await _speechRepository.GetByIdAsync(entity.SpeechId);

    var newFileName = $"{speech.PerformanceId}_{entity.SpeechId}_{entity.LanguageId}.mp3";
    var oldPath = Path.Combine(_audioFilesFolderPath, entity.FileName);
    var newPath = Path.Combine(_audioFilesFolderPath, newFileName);
    File.Move(oldPath, newPath);

    entity.FileName = newFileName;
```

```
            return entity;
        }

        private async Task<Audio> ChangeDuration(Audio entity)
        {
            var speech = await _speechRepository.GetByIdAsync(entity.SpeechId);

            var file = TagLib.File.Create(_audioFilesFolderPath + entity.FileName);
            var duration = file.Properties.Duration;
            entity.Duration = Convert.ToInt32(duration.TotalSeconds);

            if (speech.Duration >= entity.Duration)
                return entity;

            var newSpeech = new Speech {Id = speech.Id, Duration = entity.Duration};

            await _speechRepository.UpdateFieldAsync(newSpeech, "Duration");

            return entity;
        }
    }
}
```

Administration/Core/Services/GenericService.cs

```
using System;
using System.Collections.Generic;
using System.Linq.Expressions;
using System.Threading.Tasks;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Interfaces;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Services
{
    internal abstract class GenericService<T>
        where T : BaseEntity
    {
        protected readonly IRepository<T> Repository;

        protected GenericService(IRepository<T> repository)
        {
            Repository = repository;
        }

        public virtual async Task<List<T>> GetAllAsync()
        {
            return await Repository.ListAllAsync();
        }

        public virtual async Task<T> GetByIdAsync(int id)
        {
            return await Repository.GetByIdAsync(id);
        }

        public virtual async Task CreateAsync(T entity)
        {
            entity.Id = default(int);

            await Repository.AddAsync(entity);
        }

        public virtual async Task UpdateAsync(int id, T newEntity)
        {
            var oldEntity = await Repository.GetByIdAsync(id);

            if (oldEntity == null)
                throw new Exception($"{typeof(T)} entity with ID: {id} doesn't exist.");

            await Repository.UpdateAsync(oldEntity, newEntity);
        }

        public virtual async Task DeleteAsync(int id)
        {
            var entity = await Repository.GetByIdAsync(id);

            if (entity == null)
                return;

            await Repository.DeleteAsync(entity);
        }
    }
}
```

Administration/Core/Services/LanguageService.cs

```
using System.Linq;
using System.Threading.Tasks;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Interfaces;
```

```
namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Services
{
    internal class LanguageService : GenericService<Language>, ILanguageService
    {
        private readonly IAudioService _audioService;

        public LanguageService(IRepository<Language> repository, IAudioService audioService)
            : base(repository)
        {
            _audioService = audioService;
        }

        public override async Task DeleteAsync(int id)
        {
            var language = await Repository.GetByIdWithChildrenAsync(id, "Audios");

            if (language == null)
                return;

            var namesList = language.Audios.Select(audio => audio.FileName).AsEnumerable();

            _audioService.DeleteAudioFiles(namesList);

            await Repository.DeleteAsync(language);
        }
    }
}
```

Administration/Core/Services/PerformanceService.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Interfaces;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Services
{
    internal class PerformanceService : GenericService<Performance>, IPerformanceService
    {
        private readonly IAudioService _audioService;
        private readonly ISpeechService _speechService;
        private readonly ILanguageService _languageService;

        public PerformanceService(
            IRepository<Performance> repository,
            IAudioService audioService,
            ISpeechService speechService,
            ILanguageService languageService)
            : base(repository)
        {
            _audioService = audioService;
            _speechService = speechService;
            _languageService = languageService;
        }

        public async Task<List<Speech>> GetChildrenByIdAsync(int id)
        {
            var performance = await Repository.GetByIdWithChildrenAsync(id, "Speeches");

            return performance?.Speeches.ToList();
        }

        public override async Task DeleteAsync(int id)
        {
            var performance = await Repository.GetByIdWithChildrenAsync(id, "Speeches");

            if (performance == null)
                return;

            foreach (var speech in performance.Speeches)
            {
                await _audioService.DeleteAsync(speech.Id);
            }

            await Repository.DeleteAsync(performance);
        }

        public async Task<List<Language>> GetLanguagesAsync(int id)
        {
            var speeches = await GetChildrenByIdAsync(id);

            var speech = speeches.First();

            var audios = await _speechService.GetChildrenByIdAsync(speech.Id);

            var languagesIds = audios.Select(audio => audio.LanguageId).ToList();
```

```
            var listOfLanguages = new List<Language>();

            foreach (var langId in languagesIds)
            {
                listOfLanguages.Add(await _languageService.GetByIdAsync(langId));
            }

            return listOfLanguages;
        }
    }
}
```

Administration/Core/Services/SpeechService.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Interfaces;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Services
{
    internal class SpeechService : GenericService<Speech>, ISpeechService
    {
        private readonly IAudioService _audioService;

        public SpeechService(IRepository<Speech> repository, IAudioService audioService)
            : base(repository)
        {
            _audioService = audioService;
        }

        public async Task<List<Audio>> GetChildrenByIdAsync(int id)
        {
            var speech = await Repository.GetByIdWithChildrenAsync(id, "Audios");

            return speech?.Audios.ToList();
        }

        public override async Task DeleteAsync(int id)
        {
            var speech = await Repository.GetByIdWithChildrenAsync(id, "Audios");

            if (speech == null)
                return;

            await _audioService.DeleteAsync(id);

            await Repository.DeleteAsync(speech);
        }
    }
}
```

Administration/Core/AdministrationService.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using AutoMapper;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.DTOs;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Interfaces;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core
{
    internal class AdministrationService : IAdministrationService
    {
        private readonly IPerformanceService _performanceService;
        private readonly ISpeechService _speechService;
        private readonly IAudioService _audioService;
        private readonly ILanguageService _languageService;
        private readonly IMapper _mapper;

        public AdministrationService(
            IPerformanceService performanceService,
            ISpeechService speechService,
            IAudioService audioService,
            ILanguageService languageService,
            IMapper mapper)
        {
            _performanceService = performanceService;
            _speechService = speechService;
            _audioService = audioService;
            _languageService = languageService;
            _mapper = mapper;
        }

        public async Task<List<PerformanceDTO>> GetAllPerformancesAsync()
        {
            var performances = await _performanceService.GetAllAsync();
```

```
    return _mapper.Map<List<Performance>, List<PerformanceDTO>>(performances);
}

public async Task<List<SpeechDTO>> GetAllSpeechesAsync()
{
    var speeches = await _speechService.GetAllAsync();

    return _mapper.Map<List<Speech>, List<SpeechDTO>>(speeches);
}

public async Task<List<AudioDTO>> GetAllAudiosAsync()
{
    var audios = await _audioService.GetAllAsync();

    return _mapper.Map<List<Audio>, List<AudioDTO>>(audios);
}

public async Task<List<LanguageDTO>> GetAllLanguagesAsync()
{
    var languages = await _languageService.GetAllAsync();

    return _mapper.Map<List<Language>, List<LanguageDTO>>(languages);
}

public async Task<PerformanceDTO> GetPerformanceByIdAsync(int id)
{
    var performance = await _performanceService.GetByIdAsync(id);

    return _mapper.Map<Performance, PerformanceDTO>(performance);
}

public async Task<SpeechDTO> GetSpeechByIdAsync(int id)
{
    var speech = await _speechService.GetByIdAsync(id);

    return _mapper.Map<Speech, SpeechDTO>(speech);
}

public async Task<AudioDTO> GetAudioByIdAsync(int id)
{
    var audio = await _audioService.GetByIdAsync(id);

    return _mapper.Map<Audio, AudioDTO>(audio);
}

public async Task<LanguageDTO> GetLanguageByIdAsync(int id)
{
    var language = await _languageService.GetByIdAsync(id);

    return _mapper.Map<Language, LanguageDTO>(language);
}

public async Task CreatePerformanceAsync(PerformanceDTO performanceDTO)
{
    var performance = _mapper.Map<PerformanceDTO, Performance>(performanceDTO);

    await _performanceService.CreateAsync(performance);

    performanceDTO.Id = performance.Id;
}

public async Task CreateSpeechAsync(SpeechDTO speechDTO)
{
    var speech = _mapper.Map<SpeechDTO, Speech>(speechDTO);

    await _speechService.CreateAsync(speech);

    speechDTO.Id = speech.Id;
}

public async Task CreateAudioAsync(AudioDTO audioDTO)
{
    var audio = _mapper.Map<AudioDTO, Audio>(audioDTO);

    await _audioService.CreateAsync(audio);

    audioDTO.Id = audio.Id;
}

public async Task CreateLanguageAsync(LanguageDTO languageDTO)
{
    var language = _mapper.Map<LanguageDTO, Language>(languageDTO);

    await _languageService.CreateAsync(language);

    languageDTO.Id = language.Id;
}
```

```
public async Task UpdatePerformanceAsync(int id, PerformanceDTO performanceDTO)
{
    var performance = _mapper.Map<PerformanceDTO, Performance>(performanceDTO);

    await _performanceService.UpdateAsync(id, performance);
}

public async Task UpdateSpeechAsync(int id, SpeechDTO speechDTO)
{
    var speech = _mapper.Map<SpeechDTO, Speech>(speechDTO);

    await _speechService.UpdateAsync(id, speech);
}

public async Task UpdateAudioAsync(int id, AudioDTO audioDTO)
{
    var audio = _mapper.Map<AudioDTO, Audio>(audioDTO);

    await _audioService.UpdateAsync(id, audio);
}

public async Task UpdateLanguageAsync(int id, LanguageDTO languageDTO)
{
    var language = _mapper.Map<LanguageDTO, Language>(languageDTO);

    await _languageService.UpdateAsync(id, language);
}

public async Task DeletePerformanceAsync(int id)
{
    await _performanceService.DeleteAsync(id);
}

public async Task DeleteSpeechAsync(int id)
{
    await _speechService.DeleteAsync(id);
}

public async Task DeleteLanguageAsync(int id)
{
    await _languageService.DeleteAsync(id);
}

public async Task DeleteAudio(int id)
{
    await _audioService.DeleteAsync(id);
}

public void DeleteAudioFiles(IEnumerable<string> fileNames)
{
    _audioService.DeleteAudioFiles(fileNames);
}

public async Task DeleteFileAsync(int id)
{
    await _audioService.DeleteFileAsync(id);
}

public async Task<List<LanguageDTO>> GetLanguagesByPerformanceIdAsync(int id)
{
    var languages = await _performanceService.GetLanguagesAsync(id);

    return _mapper.Map<List<Language>, List<LanguageDTO>>(languages);
}

public async Task<List<SpeechDTO>> GetSpeechesByPerformanceIdAsync(int id)
{
    var listOfSpeeches = await _performanceService.GetChildrenByIdAsync(id);

    return _mapper.Map<List<Speech>, List<SpeechDTO>>(listOfSpeeches);
}

public async Task<List<AudioDTO>> GetAudiosBySpeechIdAsync(int id)
{
    var listOfAudios = await _speechService.GetChildrenByIdAsync(id);

    return _mapper.Map<List<Audio>, List<AudioDTO>>(listOfAudios);
}

public async Task UploadAudioAsync(AudioFileDTO audioFileDTO)
{
    var audio = _mapper.Map<AudioFileDTO, Audio>(audioFileDTO);

    await _audioService.UploadAsync(audio, audioFileDTO);
}

public async Task UploadWaitingAudioAsync(AudioFileDTO audioFileDTO)
{
```

```
            var audio = _mapper.Map<AudioFileDTO, Audio>(audioFileDTO);

            audio.FileName = "waiting.mp3";

            await _audioService.UploadAsync(audio, audioFileDTO);
        }
    }
}
```

Administration/Core/IAdministrationService.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.DTOs;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Core
{
    public interface IAdministrationService
    {
        Task<List<PerformanceDTO>> GetAllPerformancesAsync();

        Task<List<SpeechDTO>> GetAllSpeechesAsync();

        Task<List<AudioDTO>> GetAllAudiosAsync();

        Task<List<LanguageDTO>> GetAllLanguagesAsync();

        Task<PerformanceDTO> GetPerformanceByIdAsync(int id);

        Task<SpeechDTO> GetSpeechByIdAsync(int id);

        Task<AudioDTO> GetAudioByIdAsync(int id);

        Task<LanguageDTO> GetLanguageByIdAsync(int id);

        Task CreatePerformanceAsync(PerformanceDTO performanceDTO);

        Task CreateSpeechAsync(SpeechDTO speechDTO);

        Task CreateAudioAsync(AudioDTO audioDTO);

        Task CreateLanguageAsync(LanguageDTO languageDTO);

        Task UpdatePerformanceAsync(int id, PerformanceDTO performanceDTO);

        Task UpdateSpeechAsync(int id, SpeechDTO speechDTO);

        Task UpdateAudioAsync(int id, AudioDTO audioDTO);

        Task UpdateLanguageAsync(int id, LanguageDTO languageDTO);

        Task DeletePerformanceAsync(int id);

        Task DeleteSpeechAsync(int id);

        Task DeleteLanguageAsync(int id);

        Task DeleteAudio(int id);

        Task DeleteFileAsync(int id);

        void DeleteAudioFiles(IEnumerable<string> fileNames);

        Task<List<LanguageDTO>> GetLanguagesByPerformanceIdAsync(int id);

        Task<List<SpeechDTO>> GetSpeechesByPerformanceIdAsync(int id);

        Task<List<AudioDTO>> GetAudiosBySpeechIdAsync(int id);

        Task UploadAudioAsync(AudioFileDTO audioFileDTO);

        Task UploadWaitingAudioAsync(AudioFileDTO audioFileDTO);
    }
}
```

Administration/Core/InternalVisibleTo.cs

```
using System.Runtime.CompilerServices;

[assembly: InternalsVisibleTo("Web")]
[assembly: InternalsVisibleTo("Infrastructure")]
[assembly: InternalsVisibleTo("UnitTests")]
```

Administration/Infrastructure/Cofiguration/ IInfrastructureServiceCollection.cs

```
using Microsoft.Extensions.DependencyInjection;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Infrastructure.Configuration
{
    public interface IInfrastructureServiceCollection
    {
```

```
        void RegisterDependencies(IServiceCollection services);
    }
}
```

Administration/Infrastructure/Cofiguration/ InfrastructureServiceCollection.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Interfaces;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Infrastructure.Database;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Infrastructure.FileSystem;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Infrastructure.Configuration
{
    public class InfrastructureServiceCollection : IInfrastructureServiceCollection
    {
        public void RegisterDependencies(IServiceCollection services)
        {
            const string connection = "Data Source=dubbing.db";

            services.AddDbContext<DubbingContext>(options =>
                options.UseSqlite(connection, b => b.MigrationsAssembly("Web")));

            services.AddScoped<DbContext, DubbingContext>();

            services.AddScoped(typeof(IRepository<>), typeof(Repository<>));

            services.AddScoped<IFileSystemRepository, FileSystemRepository>();
        }
    }
}
```

Administration/Infrastructure/Database/DatabaseContext.cs

```
using Microsoft.EntityFrameworkCore;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Infrastructure.Database
{
    internal partial class DubbingContext : DbContext
    {
        public DbSet<Performance> Performances { get; set; }

        public DbSet<Audio> Audio { get; set; }

        public DbSet<Language> Languages { get; set; }

        public DbSet<Speech> Speeches { get; set; }

        public DubbingContext(DbContextOptions<DubbingContext> options)
            : base(options)
        {
        }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlite("Data Source=dubbing.db");
        }
    }
}
```

Administration/Infrastructure/Database/ModelsConfigurations.cs

```
using Microsoft.EntityFrameworkCore;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Infrastructure.Database
{
    internal partial class DubbingContext
    {
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Performance>()
                .HasMany(p => p.Speeches)
                .WithOne(p => p.Performance)
                .OnDelete(DeleteBehavior.Cascade)
                .IsRequired();

            modelBuilder.Entity<Performance>()
                .HasKey(p => p.Id);

            modelBuilder.Entity<Performance>()
                .Property(p => p.Title)
                .IsRequired();

            modelBuilder.Entity<Performance>()
                .Property(p => p.Description)
                .IsRequired();

            modelBuilder
                .Entity<Speech>()
```

```
                .HasMany(s => s.Audios)
                .WithOne(s => s.Speech)
                .OnDelete(DeleteBehavior.Cascade)
                .IsRequired();

            modelBuilder.Entity<Speech>()
                .HasKey(s => s.Id);

            modelBuilder.Entity<Speech>()
                .Property(s => s.Order)
                .IsRequired();

            modelBuilder.Entity<Speech>()
                .Property(s => s.Text)
                .IsRequired();

            modelBuilder.Entity<Speech>()
                .Property(s => s.Duration)
                .IsRequired();

            modelBuilder.Entity<Language>()
                .HasMany(l => l.Audios)
                .WithOne(a => a.Language)
                .OnDelete(DeleteBehavior.Cascade)
                .IsRequired();

            modelBuilder.Entity<Language>()
                .HasKey(l => l.Id);

            modelBuilder.Entity<Language>()
                .Property(l => l.Name)
                .IsRequired();

            modelBuilder.Entity<Audio>()
                .Property(a => a.FileName)
                .IsRequired();

            modelBuilder.Entity<Audio>()
                .Property(a => a.OriginalText)
                .IsRequired();
        }
    }
}
```

Administration/Infrastructure/Database/Repository.cs

```
using System;
using System.Collections.Generic;
using System.Linq.Expressions;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Interfaces;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Infrastructure.Database
{
    internal class Repository<T> : IRepository<T>
        where T : BaseEntity
    {
        private readonly DubbingContext _dbContext;

        public Repository(DubbingContext dbContext)
        {
            _dbContext = dbContext;
        }

        public async Task<T> GetByIdAsync(int id)
        {
            return await _dbContext
                .Set<T>()
                .FindAsync(id);
        }

        public async Task<T> GetByIdWithChildrenAsync(int id, string childrenName)
        {
            return await _dbContext
                .Set<T>()
                .Include(childrenName)
                .FirstOrDefaultAsync(e => e.Id == id);
        }

        public async Task<List<T>> ListAllAsync()
        {
            return await _dbContext
                .Set<T>()
                .ToListAsync();
        }

        public async Task AddAsync(T entity)
```

```
        {
            _dbContext
                .Set<T>()
                .Add(entity);

            await _dbContext
                .SaveChangesAsync();
        }

        public async Task UpdateAsync(T oldEntity, T newEntity)
        {
            _dbContext
                .Entry(oldEntity)
                .CurrentValues
                .SetValues(newEntity);

            await _dbContext
                .SaveChangesAsync();
        }

        public async Task UpdateFieldAsync(T entity, string fieldName)
        {
            _dbContext
                .Entry(entity)
                .Property(fieldName)
                .IsModified = true;

            await _dbContext.SaveChangesAsync();
        }

        public async Task DeleteAsync(T entity)
        {
            _dbContext
                .Set<T>()
                .Remove(entity);

            await _dbContext
                .SaveChangesAsync();
        }
    }
}
```

Administration/Infrastructure/FileSystem/FileSystemRepository.cs

```
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Entities;
using SoftServe.ITAcademy.BackendDubbingProject.Administration.Core.Interfaces;

namespace SoftServe.ITAcademy.BackendDubbingProject.Administration.Infrastructure.FileSystem
{
    internal class FileSystemRepository : IFileSystemRepository
    {
        public async Task WriteToFileSystemAsync(Audio audio, string path)
        {
            await File.WriteAllBytesAsync(path, audio.AudioFile);
        }

        public void Delete(string folderPath, IEnumerable<string> names)
        {
            Parallel.ForEach(names, name =>
            {
                var pathToAudio = Path.Combine(folderPath, name);

                File.Delete(pathToAudio);
            });
        }
    }
}
```

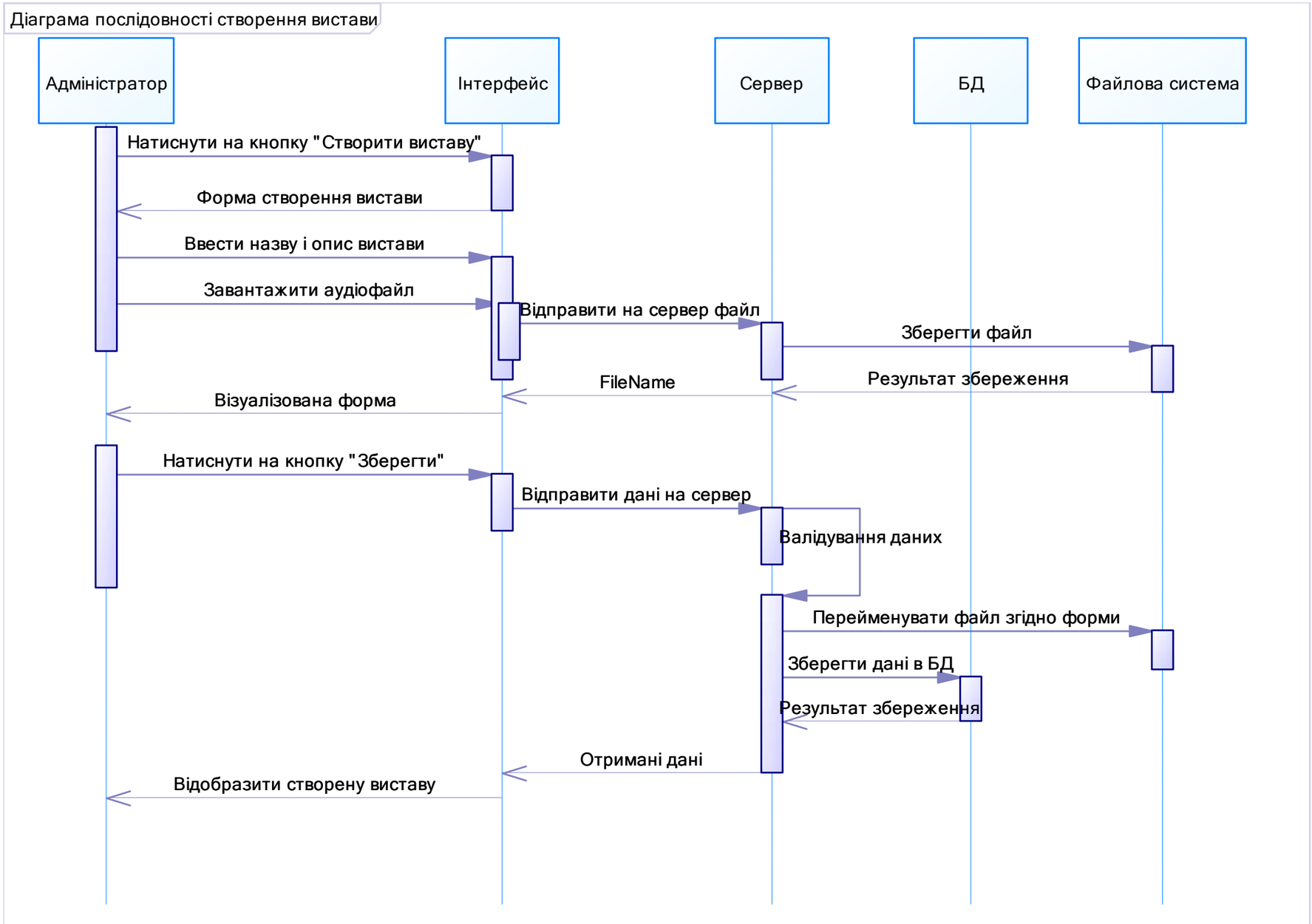
Графічний матеріал
до дипломного проекту

на тему: Система підтримки синхронного дубляжу вистав

Київ – 2019 року

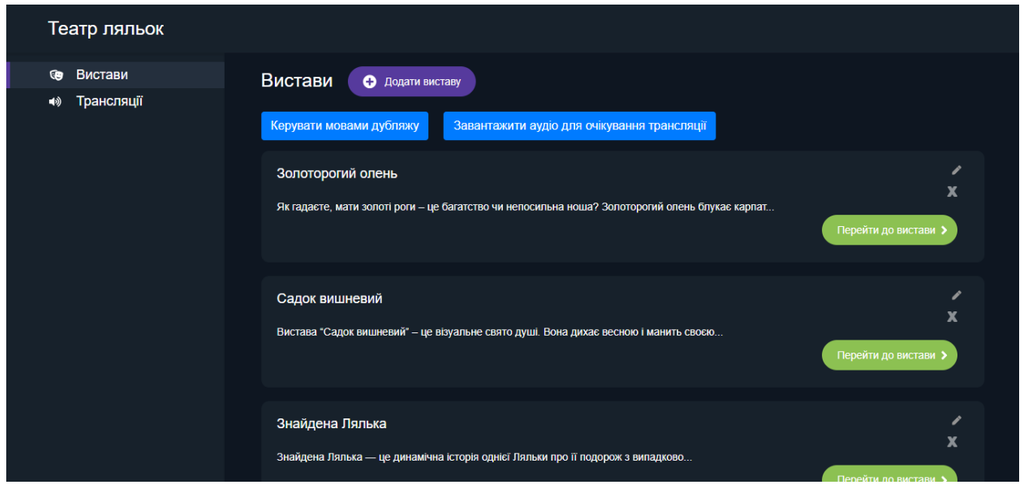
					ДП ІС-5111.1181-с.ССК	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Кас'янчк А.С				
Перевірив		Гриша О.В.				Аркуш 1 / Аркушів 2
Т. кон.					Система підтримки синхронного дубляжу вистав	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51
Н. кон.		Тєлишева Т.О.				
Затвердив		Гриша О.В.				

					ДП ІС-5111.1181-с.ССК	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Кас'янчк А.С				
Перевірив		Гриша О.В.				Аркуш 2 / Аркушів 2
Т. кон.					Система підтримки синхронного дубляжу вистав	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51
Н. кон.		Тєлишева Т.О.				
Затвердив		Гриша О.В.				

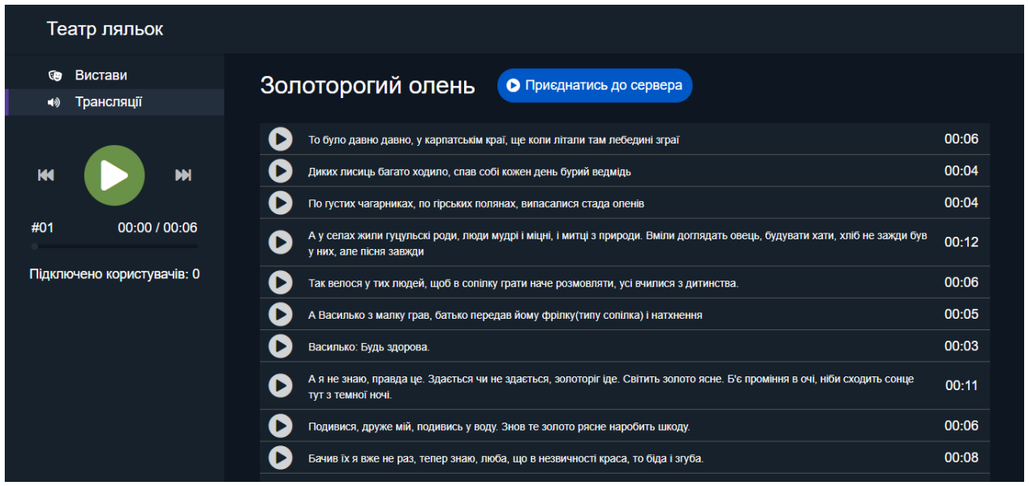


					ДП ІС-5111.1181-с.ССП	
					Схема структурна послідовності	Літера: — / Маса: — / Масштаб: —
Зм.	Арк.	№ документа	Підпис	Дата		Аркуш 1 / Аркушів 1
Розробив		Кас'янчук А.С.				
Перевірив		Гриша О.В.				
Т. кон.						
Н. кон.		Тєлишева Т.О.			Система підтримки синхронного дубляжу вистав	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51
Затвердив		Гриша О.В.				

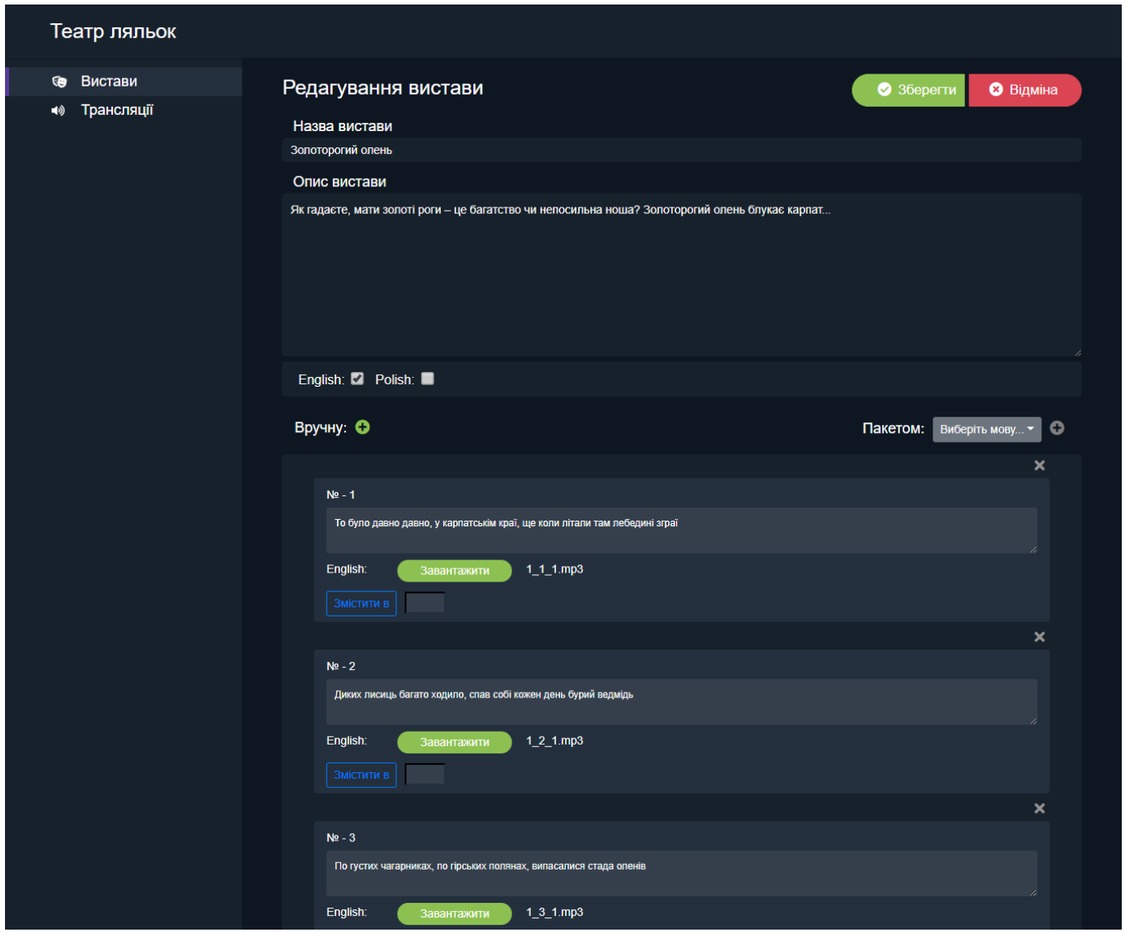
Інтерфейс початкової сторінки



Інтерфейс сторінки керування виставою

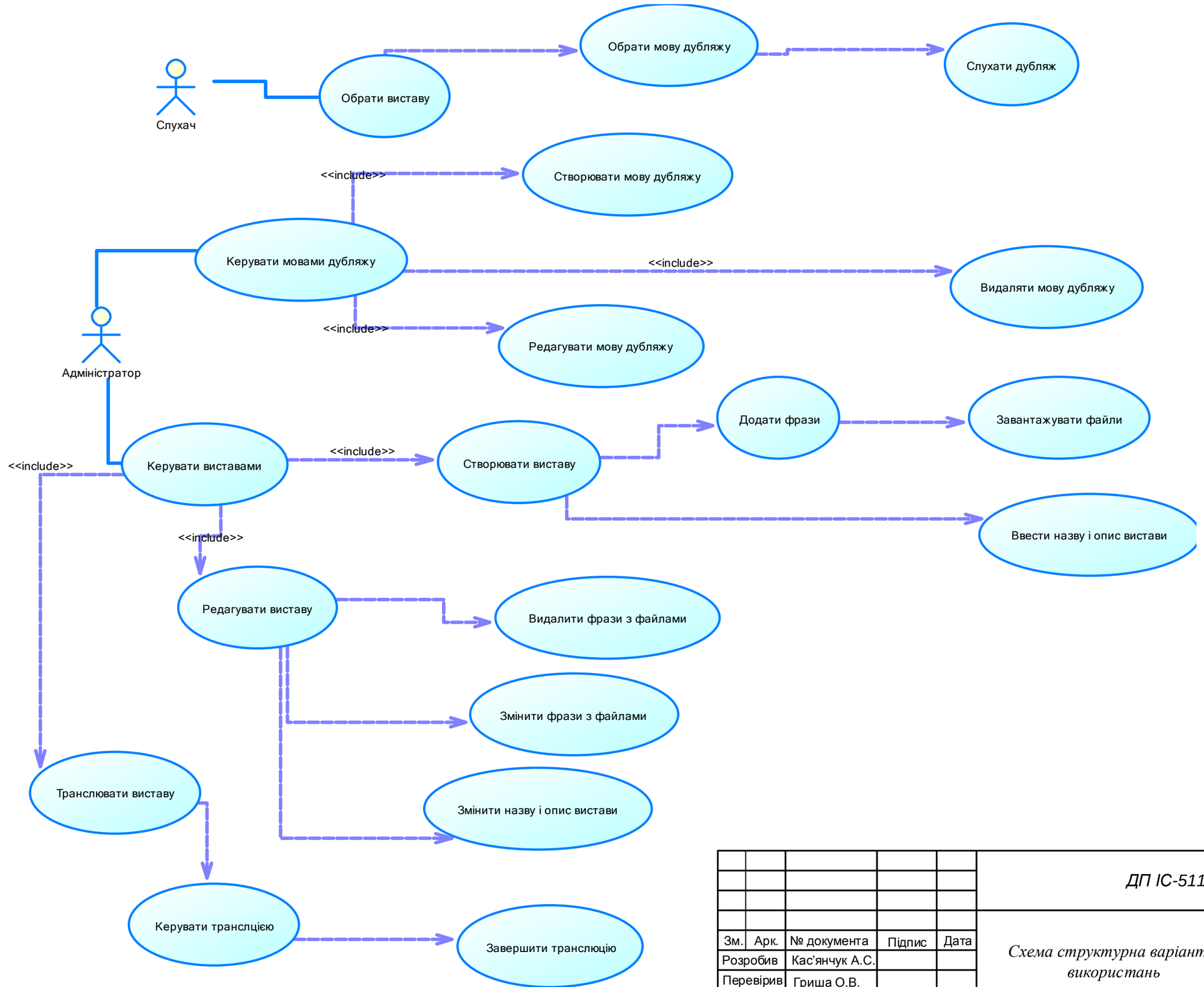


Інтерфейс сторінки редагування вистави

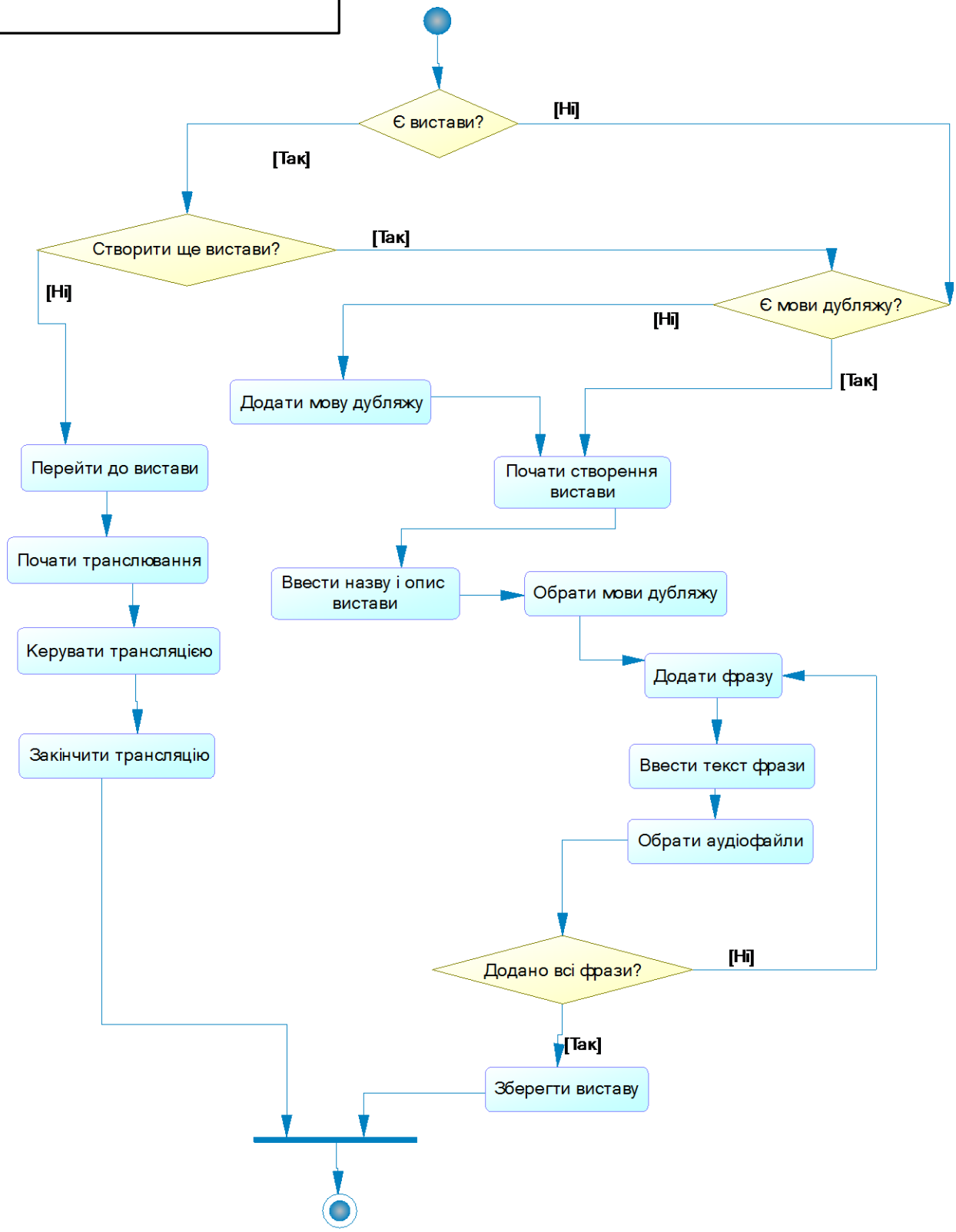


					ДП ІС-5111.1181-с. КЕ		
						Літера	Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду екранних форм		
Розробив		Кас'янчук А.С.				Аркуш 1	Аркушів 1
Перевірив		Гриша О.В.					
Т. кон.							
					Система підтримки синхронного дубляжу вистав	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51	
Н. кон.		Тєлишева Т.О.					
Затвердив		Гриша О.В.					

ДП ІС-5111.1181-с.ССВ



					ДП ІС-5111.1181-с.ССВ	
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна варіантів використань*	Літера / Маса / Масштаб
Розробив		Кас'янчук А.С.				
Перевірив		Гриша О.В.				Аркуш 1 / Аркушів 1
Т. кон.						
Н. кон.		Тєлишева Т.О.			*Система підтримки синхронного дубляжу вистав*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51*
Затвердив		Гриша О.В.				



					ДП ІС-5111.1181-с.ССС	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна станів системи	Літера / Маса / Масштаб
Розробив		Кас'янчк А.С				
Перевірив		Гриша О.В.				Аркуш 1 / Аркушів 1
Т. кон.					Система підтримки синхронного дубляжу вистав	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51
Н. кон.		Тєлишева Т.О.				
Затвердив		Гриша О.В.				

Рішення з математичного забезпечення

Дано:

- Інтенсивність приходу глядачів в театр (задається у вигляді функції);
- Максимальна допустима довжина черги – $qmax$;
- Витрати на одній пропускний пункт – p;
- Поточна довжина черги i-го пропускного пункту – q_i.

Змінні:

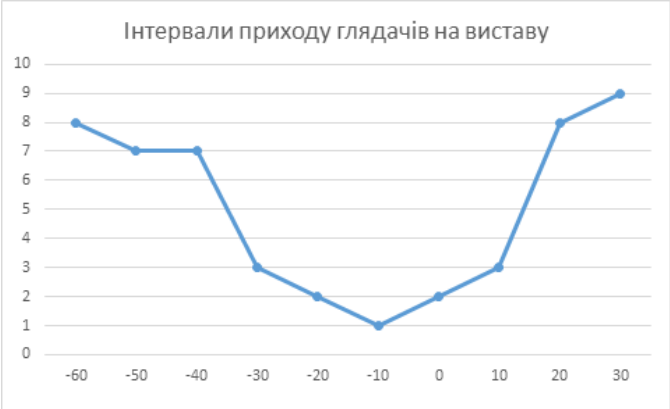
- Кількість пропускних пунктів – n.

Цільова функція – мінімізувати витрати на пропускні пункти:

$$n * p \rightarrow min \qquad (3.1)$$

Обмеження:

$$q_i \leq qmax \qquad (3.2)$$



Інтенсивність задається функцією. Вісь x задає час до початку вистави помножений на -1. Вісь y задає кількість нових глядачів.

Аналіз результатів

№	Кількість пропускних пунктів	Максимальна довжина черги	Кількість експериментів
1	1	10	50
2	2	4	50
3	3	3	50
4	4	3	50

За результатами експериментів зрозуміло, що оптимальною кількістю пропускних пунктів є 3.

Демонстраційний плакат до дипломного проекту
«Система підтримки синхронного дубляжу вистав»

Виконав студент гр. ІС-51 *Кас'янчук А.С.*
Керівник ДП *Гриша О.В.*

ДП ІС-5111.1181-с.СБД

					ДП ІС-5111.1181-с.СБД	
Зм.	Арк.	№ документа	Підпис	Дата	Схема бази даних	Літера / Маса / Масштаб
Розробив		Кас'янчк А.С				
Перевірив		Гриша О.В.				Аркуш 1 / Аркушів 1
Т. кон.					Система підтримки синхронного дубляжу вистав	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51
Н. кон.		Тєлишева Т.О.				
Затвердив		Гриша О.В.				

ДП ІС-0203.1393-с.ССД

					ДП ІС-0203.1393-с.ССД		
Зм.	Арк.	№ документа	Підпис	Дата	Назва схеми	Літера	Маса / Масштаб
Розробив		Кас'янчк А.С					
Перевірив		Гриша О.В.				Аркуш 1	Аркушів 1
Т. кон.					Система підтримки синхронного дубляжу вистав	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-41	
Н. кон.		Тєлишева Т.О.					
Затвердив		Гриша О.В.					